A PRIMERA REVISTA DE PROGRAMACION EN CASTELLANO

SOLO PROGRAMADORES

O VI. 2ª ÉPOCA NÚMERO 61 UNA PUBLICACIÓN DE: REVISTAS PROFESIONALES S.L. 975 Ptas. • 1.280 Esc-Cont • 5,86 € (IVA incluido)

INUX: DNS A FONDO (II)
ómo obtener información

MULTIMEDIA
reación de un reproductor multimedia en Delphi IV

A TECNOLOGÍA ASP (II)
l lenguaje JScript

BDD: DESARROLLO CLIENTE/SERVIDOR (Y V)
reación de BBDD en SQL Server

JAVA: ENCRIPTACIÓN (II)
Generación de claves

VB: SUITE DE INTERNET (III)
El protocolo POP3

WINDOWS NT
Las características de IIS 4.0

TRANSFERENCIA DE ARCHIVOS PUNTO A PUNTO (IV)
Implementación del protocolo de enlace

API de telefonía (TAPI)

CD



• Perl Development Kit 1.2.3 • Entisoft Tools 1.5
• ClassMapper 2.0 • Sheriff SDK 2.1 • HTML Transit 4.0
• Component Builder 3.0 • ColdFusion Express 4.0



SÓLO PROGRAMADORES

Número 61
SÓLO PROGRAMADORES
es una publicación de
REVISTAS PROFESIONALES S.L.

Editor
Agustín G. Buelta
Director
Javier Amado Buiza
Coordinador Técnico
Eduardo De Riquer Frutos
Coordinadoras de Redacción
Gema Romero Moreno-Manzanaro
Cristina Peña del Pozo
Colaboradores
Constantino Sánchez, Juan Luis Ceada, Jorge Delgado, Javier Sanz, Adolfo Aladro, Javier Toledo, Esteban Amado, Jordi Agost, Vicente A. Sánchez Werner
Maquetación y Tratamiento de Imagen
J. Santos, Alfonso Sabán
Consultas técnicas
atecnica@virtualsw.es

Publicidad
Paloma Seidel
Tel.: (91) 304 78 46
Mariano Sánchez (Barcelona)
Tel.: (93) 322 12 38
Pepín Gallardo (Barcelona)
Tel.: 617 09 36 68

Suscripciones
Rosa Tabares
Tel. (91) 304 87 64 Fax: (91) 327 13 03

Preimpresión
Grebe, S.L.
Impresión
I.G.Pantone
Distribución
Motorpress Ibérica

Depósito legal: M-26827-1994
ISSN: 1134-4792
PRINTED IN SPAIN

Precio en Canarias, Ceuta y Melilla: 938 ptas. sin I.V.A.
Con sobretasa aérea: 975 ptas.sin I.V.A.

EDITORIAL

Menú especial para todos

Un número más nos apuntamos a las nuevas tecnologías y ofrecemos en nuestro reportaje de portada los métodos y modos de poder gestionar dentro del mundo de la programación la telefonía. Más concretamente la gestión de la línea telefónica, por medio de la cual hoy día son enviados la mayoría de los datos con los que trabajamos. Y las perspectivas de futuro indican que cada vez más es y será así.

El mundo multimedia desde que apareció cautivó a muchos usuarios y poco a poco ha ganado más adeptos y adictos. Para todos aquellos programadores de *Delphi* (que sabemos de buena tinta que no sois pocos), en la sección de multimedia mostramos cómo programar un reproductor. Gracias a él y mediante pasos muy sencillos podemos llegar a crear un reproductor que soporte la mayoría de los formatos que actualmente existen. Gracias a los controles *ActiveX* podemos reproducir formatos como *.WAV* y con otros incluso el tan conocido *MP3*. Un verdadero reto que a muchos les merecerá la pena aprovechar y dedicar un rato para crear un propio reproductor personalizado, todo ello en *Delphi*.

Y como no, para los que el mes pasado se quedaron con los dientes largos para proseguir alguna de las secciones que este mes continúan, ofrecemos las siguientes partes de encriptación en *Java*, cómo crear tu propia Suite de *Internet* en *Visual Basic*, Bases de datos Cliente/Servidor, transferencia de archivos punto a punto, y mucho más.

Además, para aquellos que despertó su atención *ASP* en el número anterior, este mes profundizamos un poco más ofreciendo entre otros temas un análisis sobre el lenguaje *JScript* para los usuarios más exigentes.

Tampoco podía faltar, el espacio que reservamos el mes pasado, como sección fija, para todos los amantes de *Linux*. Este mes dedicamos estas páginas a profundizar un poco más en el mundo del *DNS*.

Como podéis comprobar os ofrecemos un menú bastante completito y para todos los gustos, esperamos que una vez más degustéis las páginas como si del menú más exquisito se tratase, por lo menos los cheff de la redacción hemos puesto todo nuestro empeño en ello.

Buen provecho.

Javier Amado
DIRECTOR

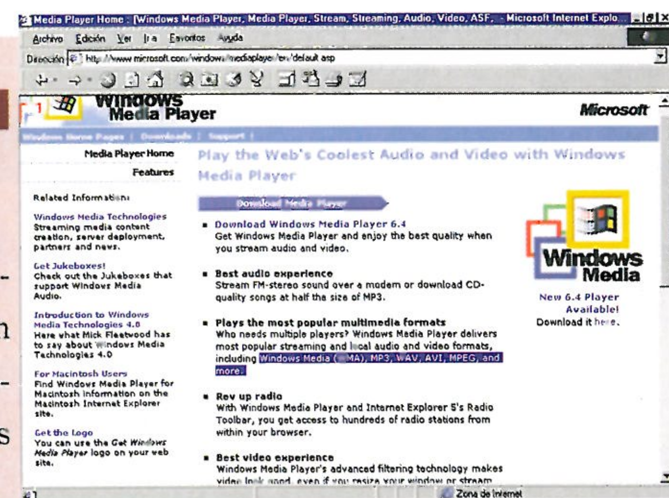


SÓLO PROGRAMADORES

22 Multimedia

PROGRAMAR UN REPRODUCTOR MULTIMEDIA CON DELPHI (I)

Para trabajar sin problemas con todo tipo de ficheros de sonido y vídeo vamos a desarrollar un reproductor multimedia utilizando como lenguaje de programación *Delphi* y las herramientas que nos proporcionan *Inprise* y *Microsoft*.



44 Java

ENCRIPTACIÓN (II)



Tras la introducción general a la criptografía y el desarrollo de los métodos clásicos más elementales que vimos en el número anterior, pasamos a ver los fundamentos de la generación de claves, con los algoritmos más importantes que forman los criptosistemas de clave privada.

54 Visual Basic

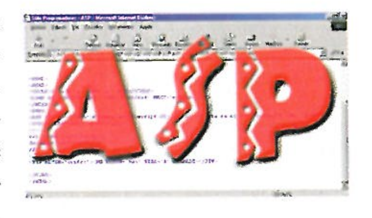
SUITE DE INTERNET (III)



En el número anterior introdujimos el correo electrónico en nuestra *suite*, ahora incorporamos a esta aplicación un cliente para que así podamos recibir mensajes de servidores. Para ello utilizaremos el protocolo *POP3*.

60 Internet

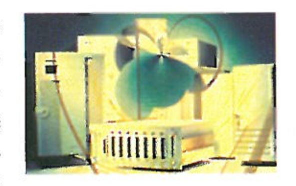
LA TECNOLOGÍA ASP (II): EL LENGUAJE JSCRIPT



En el capítulo anterior vimos los fundamentos de la programación de páginas *ASP*. En esta ocasión analizaremos el lenguaje *JScript*, que nos permite elaborar programas tan complejos como sea necesario, aunque empezaremos por lo más fácil, y después lo iremos complicando poco a poco.

66 Bases de datos

DESARROLLO CLIENTE/SERVIDOR (Y V)



Hasta ahora se han ido exponiendo los argumentos que definen las características de una aplicación cliente/servidor, desde ambos puntos de vista. Para finalizar esta serie de artículos mostramos los pasos para la creación de la base de datos en *SQL Server* y las modificaciones que ha de sufrir nuestro programa para poder utilizarla.

72 Redes

TRANSFERENCIA DE ARCHIVOS PUNTO A PUNTO (IV)

Una transmisión correcta en una comunicación se consigue detectando los posibles errores. De esta tarea se ocupa el nivel de enlace y ese nivel será el que programemos en nuestro próximo componente.

78 Libros

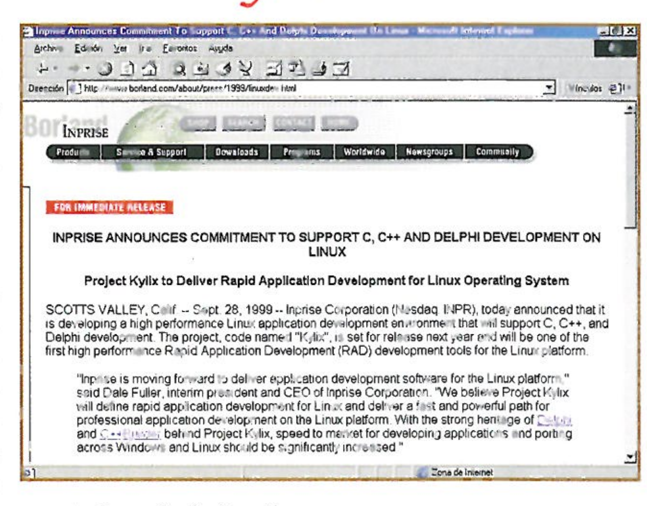
Las últimas novedades editoriales sobre programación y los libros que nos han parecido más interesantes aparecen en estas páginas, con una sencilla reseña que os ayude a conocerlo y a decidir cuál es el que más os conviene.

80 Dudas técnicas

Como siempre estamos dispuestos a contestar a todas vuestras dudas. No os preocupéis si os parecen demasiado complejas o demasiado sencillas, siempre podréis acudir a nuestra dirección de e-mail e intentaremos solucionar vuestros problemas: **solop@virtualsw.es**.

INPRISE CREARÁ APLICACIONES LINUX BASADAS EN C, C++ Y DELPHI

Inprise, empresa antes conocida como *Borland*, ha anunciado que a partir de ahora desarrollará aplicaciones *RAD* para el sistema operativo *Linux* que soportarán *C*, *C++* y *Delphi*.



Este proyecto, denominado *Kylix* va a estar muy influenciado por los resultados obtenidos por la encuesta realizada por *Borland* entre los desarrolladores *Linux*. De hecho gracias a estos datos se prevé que en un futuro este proyecto soporte las principales distribuciones *Linux*, incluidas *RedHat* y *Corel Linux* de próxima aparición. Sin embargo ese es el futuro, más actual es lo que pretenden conseguir a lo largo del próximo año, como la creación de un componente utilizable para *Internet*, bases de datos, servidor de aplicaciones e incluso para el desarrollo de la interfaz gráfica de usuario.

Además de todo lo anteriormente expuesto se incluirá un compilador nativo de alta velocidad de *C/C++/Delphi* para *Linux* y se implementará una versión *Linux* de la Librería de Componentes Visuales (*VCL*) de *Borland*.

Para más información sobre este proyecto se puede acudir a la página: **http://borland.com/about/press/1999/linuxdev.html**

CONTROLIT 5.0, LA SOLUCIÓN DE CONTROL REMOTO DE PRÓXIMA GENERACIÓN

Ya está disponible en el mercado *ControlIT 5.0*, la nueva versión de esta herramienta de control remoto, altamente escalable y de uso sencillo con la que los administradores pueden acceder y controlar *PC's* o servidores en cualquier lugar de la Red.

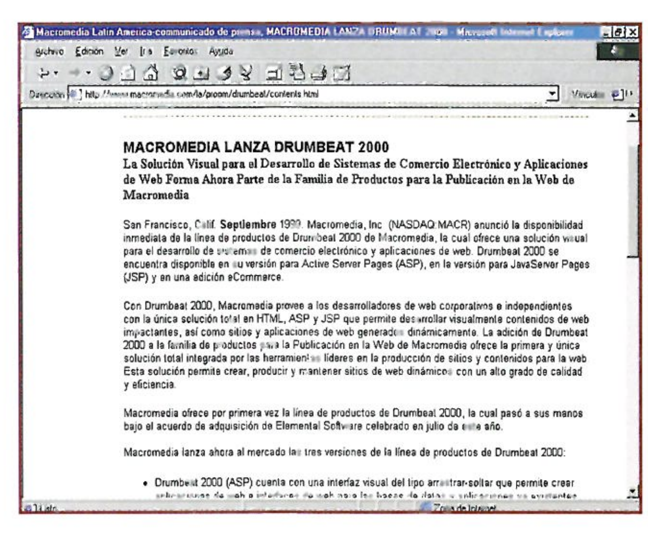
Esta nueva versión mejorada supone la solución de control remoto independiente líder del sector informático. *ControlIT 5.0* sube el listón en la utilización de funciones de control remoto en grandes entornos corporativos, simplificando el mantenimiento necesario y las actividades de gestión.

La versión 5.0 soporta *Windows* 2000 *Beta*, tanto observador como visitante. Reduce la tarea de mantenimiento de los sistemas de control remoto, permitiendo a su vez la realización de gestiones de políticas centrales. Además incluye un sólido motor de encriptación por lo que se mejora la integridad de los datos.

El inicio y la detención de las actividades de control remoto o el soporte de *HTTP* son algunas de las posibilidades que ofrece esta herramienta que ya está disponible en el mercado con un precio orientativo de 1990 dólares para licencias de dos usuarios.

DRUMBEAT 2000, LO MÁS NUEVO EN DESARROLLO WEB

A partir de ahora los usuarios de *Internet* podrán disponer de una nueva solución visual para el desarrollo de aplicaciones *Web* y de comercio electrónico. Se trata de *Drumbeat* 2000 que ya está disponible en sus versiones *Active Server Pages* (*ASP*), *e-comerce edition* y *Java Server Pages* (*JSP*). Con esta nueva herramienta la empresa *Macromedia* ofrece a los desarrolladores *Web* una solución *HTML, ASP* y *JSP* completa para el desarrollo visual de contenidos *Web*, amplicaciones y sitios *Web* generados de una manera dinámica.



Drumbeat 2000 (*ASP*) posee una interfaz *Drag-And-Drop* visual, que nos permite la creación de interfaces y aplicaciones *Web* a partir de aplicaciones y bases de datos ya existentes.

Además esta versión presenta sofisticados *Wizards* o asistentes que permiten construir aplicaciones *ASP* para acceder y actualizar en tiempo real datos que funcionan en cualquier navegador y que no requieren código manual.

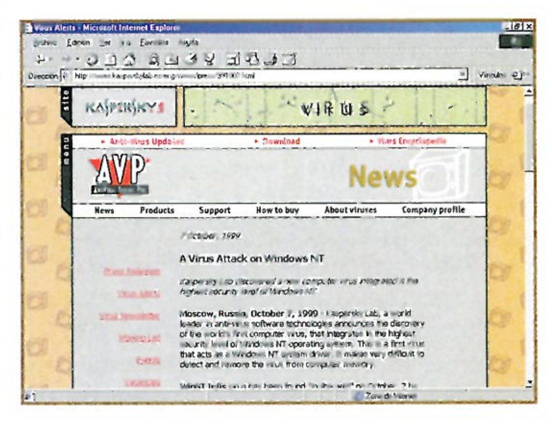
Por su parte *Drumbeat 2000* (*JSP*) crea aplicaciones que conectan con los procesos de *Backend* utilizando *JavaBeans* y *Servlets*, acelerando así el proceso de desarrollo de aplicaciones *JSP*, ya que genera *JavaScript*, *Java*, *HTML* y *JSP* libre de errores.

Para completar esta información se puede acudir a la dirección: **http//:www.macromedia.com/la/proom/drumbeat/contents.html.**

DETECTADO VIRUS EN EL NIVEL MÁS ALTO DE SEGURIDAD DE WINDOWS NT

El nivel más alto de seguridad de *Windows NT* tampoco escapa de los virus. La compañía *Kapersky Lab*, líder mundial en tecnología *software* antivirus ha anunciado la presencia de un virus integrado en este nivel.

Este virus, que actúa como un controlador de este sistema operativo, ha sido bautizado como *WinNT.Infis*, y fue detectado el 7 de octubre. Se trata de un archivo residente en memoria, por lo que es muy difícil detectarlo y eliminarlo de la memoria.



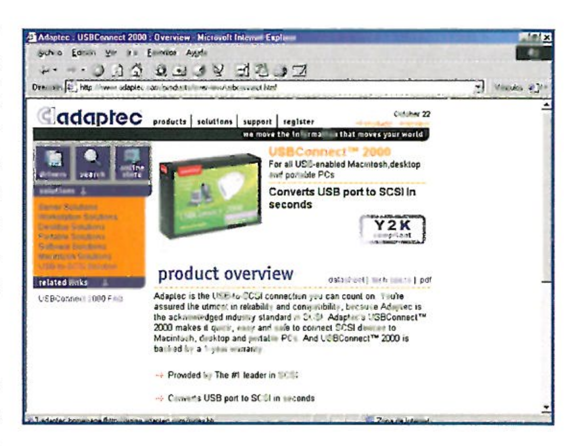
Sabremos que nuestro sistema se ha visto infectado ante la imposibilidad de ejecutar algunos programas como *MSPAINT.EXE*, *CALC.EXE*, *CDPLAYER.EXE*, etc, dado que un error de programación en el virus daña su codificación. Otro indicador es que cuando se ejecuta un archivo infectado aparece el archivo *INF.SYS* en la carpeta */Win/NT/System32/Drivers*.

Afortunadamente este virus no afecta a los sistemas bajo *Windows 95/98, Windows 2000 ...*, sólo es operativo bajo *Windows NT* 4.0 con los *Service Packs* 2, 3, 4, 5, y 6 instalados. De hecho únicamente infecta archivos *EXE PE* excepto *CDM.EXE* (*Windows NT command processor*).

La detección y eliminación del virus ha sido añadida en una actualización de urgencia del antivirus *Antiviral Toolkit Pro* (*AVP*) que está disponible en la *Web*: **http://www.avp.ru**

YA ES POSIBLE CONECTAR PERIFÉRICOS SCSI A TRAVÉS DE PUERTOS USB

A partir de ahora, y gracias a *USBConnect 2000* de *Adaptec* ya es posible tener conectados periféricos SCSI como discos duros, unidades *CD-ROM* externas, *Jazz*, escáneres... en cualquier ordenador que posea un puerto *USB*.



Este nuevo conversor aunque en un principio se dirige a los usuarios del nuevo *iMac* de *Apple* también se puede aplicar en todas las plataformas *Macintosh*, así como los *PC's* y portátiles que también dispongan de este tipo de puerto.

USBConnect que estará a la venta próximamente a través de los distribuidores habituales y en la tienda *online* de *Adaptec*, integra un cable conector directo al puerto *USB*, por lo que no requiere la apertura del equipo. Además, reconoce instantáneamente los periféricos, con lo que se evite el tener que reiniciar el sistema.

Este nuevo producto que incluye una guía de usuario y un *CD-ROM* de configuración tiene un precio recomendado de 16.100 pesetas. Si deseáis más información podéis dirigiros a: **http://www.adaptec.com/products/overview/ubsconnect.html.**

ACCESS 2000 YA SE PUEDE INTEGRAR CON SQL SERVER 7

Las nuevas características de *Access* 2000, y en concreto *Access Projetcs* permiten crear una aplicación de bases de datos con *Access* en el *front end* y una conexión nativa a *SQL Server* mediante *OLE DB* en el *back end*. Por lo que es posible transformar las aplicaciones *Access* en equivalentes a sus equivalentes en *SQL Server* 7.

A partir de ahora la interfaz de *Access* soporta la creación y ejecución de procesos y desencadenadores (*triggers*) almacenados en el *SQL Server*.

Así se reducen las barreras para la creación de aplicaciones cliente/servidor, con lo que se obtiene un mayor partido tanto a *SQL Server* como a *Access*.

Todos aquellos que estén interesados pueden obtener más informacíonen: **Http://microsoft.com**

BREVES

Mejoras en SQL Server 7.0

Los procedimientos de almacenamiento en las empresas y la distribución de datos vía Internet, más conocidos como replicaciones, se vuelven mucho más seguros combinando SQL con el servidor Proxy de Microsoft.

Los pasos a seguir incluyen la configuración de la topología de red, comprensión de la metodología de seguridad, configuración del Servidor Proxy y configuración de SQL Server 7.0 para la replicación. Los interesados en el proceso pueden encontrar un completa explicación en http:microsoft.com/technet/sql/intrepl.htm

Java vuelve a los ordenadores personales

La nueva versión de Java, que estará disponible el próximo mes de enero volverá a centrarse en los ordenadores de sobremesa, según han afirmado los responsables del proyecto desde Sun Microsystems.

El proyecto incluye una colección de componentes que debería permitir su funcionamiento en cualquier máquina compatible con Java, sin tener en cuenta el hardware. Sin embargo todavía quedan muchos problemas de funcionalidad y velocidad para que este proyecto sea una realidad. Se puede encontrar más información en http://www.sun.com/

Contenido del CD-ROM

HERRAMIENTAS DE PROGRAMACIÓN

ENTORNOS DE PROGRAMACÓN

CODEWHIZ 1.5.1.0999B

Editor de código fuente para programadores que puede ocultar o mostrar cualquier sección de código. Ofrece soporte para numerosos lenguajes como *Delphi, SQL, VB Script, JScript, C, C++, HTML...*



DATA JUNCTION 7.0

Traduce datos estructurados de 80 tipos de ficheros incluyendo bases de datos, hojas de cálculo, ficheros *ASCII*, *COBOL*, binarios, estadísticos, etc. Ofrece soporte para *ActiveX*, *XML*, *Windows* 2000 y *Office* 2000.



DIRECTX SDK 7.0

Kit de desarrollo para aplicaciones gráficas. Incluye todas las herramientas necesarias para desarrollar programas que aprovechen las ventajas de la *API* de *DirectX*. Esta actualización ofrece soporte para *DLS2* y *DirectDraw*.



PERL DEVELOPMENT KIT 1.2.3

Kit de desarrollo para *Perl* que incluye un depurador de código fuente, un generador de controles, un compilador y un generador de objetos *COM*.



WINEDIT 99E

Editor de textos para profesionales. Puede leer y escribir ficheros de texto de *Macintosh* y *UNIX*, abrir múltiples ficheros, compararlos, gestionar proyectos, etc. Incluye enlaces directos a las librerías de ayuda de los distintos lenguajes, un lenguaje de macros con más de 550 funciones y plantillas para diversos lenguajes.



LENGUAJES

VISUAL BASIC

ENTISOFT TOOLS 1.5

Colección de 1.000 rutinas para *VB* y *Microsoft Office*. Las rutinas permiten una mejor manipulación de cadenas, almacenamiento de datos, desarrollo *software* y otras funciones que se utilizan en programas de *VB* y aplicaciones que soporten este lenguaje.

EXCALIBUR RAINBOWBUTTON 1.0

Control *ActiveX* para *Visual Basic* 6.0 que mejora la apariencia de los botones de nuestras aplicaciones. Ofrece la posibilidad de utilizar formas predefinidas, crear otras nuevas, determinar su posición, los colores y gradientes, o asignar dos apariencias distintas según el estado de los botones.

FORMATVB 1.1

Facilita el mantenimiento de cualquier aplicación volviendo a dar formato al código fuente de modo que pueda ser legible. Procesa 1.000 líneas de código por segundo, tanto de ficheros concretos, como de grupos o incluso proyectos enteros. Crea automáticamente ficheros *backup* y puede volver a restaurar el estado inicial de los ficheros.

JAVA

CLASSMAPPER 2.0

Herramienta de ingeniería inversa para *Java*. Incluye herramientas de análisis visual y descompilación, así como un visor de las relaciones. Descompila automáticamente el código *Java* y crea diagramas de la estructura básica de información. Permite explorar las propiedades de cualquier clase, método o campo.

CONDENSITY 1.0.6

Protege el código Java de operaciones de ingeniería inversa. Contrae, oculta y optimiza el código, haciéndolo más pequeño y eficiente. *Condensity* analiza el código para determinar qué métodos pueden ser procesados de forma segura, y ofrece un control completo para manipularlos y obtener un buen equilibrio entre protección, tamaño y operatividad.



FREEJAVA 2.0.1

Entorno de desarrollo integrado que trabaja con *JDK* de *Sun*. Incluye un editor de código fuente, un visor de ficheros, clases y funciones, una ventana de proyecto que facilita la recopilación y apertura de los ficheros fuentes *Java* y *HTML*. Permite compilar y ejecutar aplicaciones *Java* y *applets* desde el interior del *IDE*.

HTML

CUTEMAP 1.1

Genera mapas de imágenes interactivos, creando cada área de hiperenlace, su posición, el código *HTML*, y graba todo ello en un fichero *HTML* que se añade al código de la página *Web*. Soporta todos los formatos de imágenes.

HOTDOG PAGEWIZ 1.0



Editor visual *HTML* para varios niveles, incluye asistentes y ofrece

dos modos de funcionamiento. El modo *Express* utiliza numerosas plantillas y el modo *Editor*, presenta una interfaz intuitiva para crear las páginas *HTML*.

ROBOFORM 2.0B2

Herramienta para *Internet Explorer* que facilita la introducción de datos en formularios *Web*. Primero se introduce en el programa los datos y cuando deseemos completar un formulario sólo hay que seleccionar "*ROBOFORM auto*" y el programa completará los campos que reconozca.

SURFMAP JAVASCRIPT 2.1

Diseña componentes de navegación basados en *JavaScript* para sitios *Web*. No es necesario escribir código fuente, ya que trabaja haciendo selecciones de estilo, colores de fondo, el tamaño, fuente, estilo y color del texto para cada uno de los ítems, etc.

OTROS

ANETHELPTOOL 5.0.0.15

Sistema visual de autor que crea ficheros de ayuda para *Windows* y documentos *HTML*. Presenta dos modos de trabajo, uno para crear y editar los documentos (*Design*) y el otro (*Runtime*), para introducir cambios antes de compilar, ya que emula un entorno real *WinHelp*. Puede publicar los ficheros en formato *Windows Help*, *HTML* o *RTF*.

COMPONENT BUILDER 3.0

Reduce la cantidad de código necesario para crear nuevos componentes *Delphi*. Sólo hay que describir las propiedades de los componentes, métodos y eventos, y el programa genera automáticamente la estructura y los elementos necesarios.

GRAPHINGDLL 1.2

Conjunto de librerías de rutinas y clases que pueden ser incluidas en cualquier aplicación *Windows MFC*, para mostrar datos en forma de gráficos. Incluye varios tipos como líneas, barras horizontales y verticales, gráficos circulares e histogramas.

PROHELP 1.1

Conjunto de herramientas de programación que mejoran las funciones de ayuda sensible al contexto de las aplicaciones. Trabaja sin añadir código fuente adicional y soporta todos los métodos de acceso a la ayuda sensible al contexto en un estilo "What's This?". Incorpora temas de ayuda que abarcan la mayoría de los objetos estándar.

SHERIFF SDK 2.1

Protege aplicaciones *Windows* de 32 *bits* de la copia ilegal. Proporciona varios esquemas de licencia de *software* que permiten definir aspectos como la duración de una versión de evaluación, las funciones que estarán activas, etc. Incluye clases y aplicaciones de demostración para *Visual C++*, *Visual Basic*, *Visual FoxPro* y *Delphi*, además de una librería estática y un control *ActiveX*.

HERRAMIENTAS Y UTILIDADES

ADMINISTRATOR PASSWORD RECOVERY KIT 2.1

Recupera *passwords* olvidadas o perdidas. Incluye un módulo de recuperación de claves para *Windows NT* que restaura la protección por *password* o las opciones de seguridad de inicio, cuando una clave se encuentra perdida.

ANOTHER TASK MANAGER 2.011

Gestiona las prioridades de sistema de todos los programas en ejecución. Proporciona funciones para controlar en tiempo real todos los programas en ejecución, cerrarlos, monitorizar el uso de la *CPU* y establecer los atributos de ventana.

BITMAP EXTRACTOR 1.00

Extrae recursos *bitmap* desde ficheros ejecutables *Windows* de 32 *bits*. Puede obtener recursos como *EXE, DLL, OCX*, etc. y situarlos en un directorio concreto. Además permite definir máscaras para filtrar los ficheros.

HTML TRANSIT 4.0

Convierte automáticamente documentos en páginas *Web*. Ofrece un control total sobre los elementos de navegación que han sido añadidos y sobre la apariencia global de las páginas. Convierte documentos (incluyendo gráficos, tablas objetos *OLE*) desde *Word*, *WordPerfect*, *AmiPro*, *Interleaf*, *Microsoft Write*, *RTF* y *ASCII*.

JVDE EBACKUP STANDARD 2.41

Permite encriptar y desencriptar ficheros de una manera rápida. El programa incluye tres potentes algoritmos: *Blowfish*, *CAST5*, y *DES*. Ofrece la posibilidad de enviar mensajes encriptados de correo electrónico.

RESPLENDENT REGISTRAR 1.08

Completo editor del Registro de *Windows*, que puede trabajar desde la línea de comandos o el menú contextual. Incluye opciones para realizar búsquedas y sustituciones, guardar y restablecer funciones y hacer un seguimiento de los cambios en el Registro.

WHATFORMAT 2.0

Identifica el formato de ficheros de tipo desconocido, analizando los primeros del mismo. Reconoce numerosos formatos incluyendo archivos, gráficos, multimedia, de sonido, bases de datos, etc.

REDES

COLDFUSION EXPRESS 4.0
Servidor de aplicaciones *Web* que proporciona una plataforma abierta su publicación. Soporta 15 de los *tags CFML* más populares y numerosas funciones estándar. Ha sido diseñado para trabajar con el entorno de edición *HTML Allaire HomeSite*.

I.SHARE 3.5
Permite a los usuarios de una red compartir una única conexión a *Internet*. Ofrece seguridad *firewall*, seguimiento de la actividad, conexión automática, e incluye opciones para limitar el acceso a determinados sitios *Web*.

IDEM 1.2
Automatiza la sincronización de ficheros entre dos ordenadores que se encuentren conectados en red. El programa asegura los datos importantes copiándolos en otro ordenador. Puede preservar los nombres de ficheros *Macintosh* y las estructuras guardadas en un servidor *Windows NT* 4.0 ejecutando *SFM* en particiones *NTFS*.

INVESTIGATOR 2.0
Registra la actividad de ordenadores independientes o en red. Trabaja en segundo plano de modo invisible, y registra datos como la fecha, la hora de inicio y duración de las sesiones y los programas que se han ejecutado.

MERAK 2.10.210
Servidor de correo electrónico potente, rápido y seguro. Puede trabajar con múltiples dominios, alias, listas de correo, dominios virtuales, realizar un seguimiento de la actividad, programar tareas y enviar autorespuestas. Soporta los protocolos *POP3*, *SMTP* y *ESMTP*.

RELAYFAX 2.0.1
Cliente/servidor que se integra con el sistema de correo electrónico existente para proporcionar funciones de fax utilizando la opción de imprimir de cualquier aplicación *Windows*. Está optimizado para trabajar con *WinGate* y *MDaemon*! Ofrece soporte para ficheros atachados, acceso *RAS* y detección automática de módem fax.

WEBLOG MANAGER PRO 1.0.45
Analiza la actividad de un servidor *Web* y muestra información como las estadísticas más comunes y los tipos de navegadores. Puede hacer un seguimiento de los usuarios que visitan varias veces el sitio y permite denegar el acceso a determinados visitantes utilizando como referencia su dirección *IP*.

DOCUMENTACIÓN/ TUTORIALES

10 MINUTE GUIDE TO PRACTICAL UNIX 8.0
Tutorial de *UNIX* que presenta una completa información organizada según las tareas que los usuarios deben realizar diariamente.

NEWBIES GUIDE TO PUBLISHING AN EZINE 2.0
Cuenta como publicar un boletín de correo electrónico, y proporciona los elementos que se incluirán en el mismo. Ofrece información para localizar artículos, conseguir suscriptores y anunciantes, e intercambiar anuncios con otros editores.

WEBTUTOR 3.7
Colección de tutoriales sobre creación de páginas *HTML*. Se encuentran en formato *HTML* por lo que es necesario utilizar un navegador para su consulta. Esta actualización incluye la última versión de *The Barebones HTML Guide*.

IMPRESCINDIBLES

ANTIVIRUS
AntiViral Toolkit Pro 3.0.129
McAfee VirusScan 4.0.3
Panda Antivirus 6.10. Platinum

GRÁFICOS
Icon Bank 4.0 Gold Edition Web Edition / Desktop Edition
IconForge 4.5
MicroAngelo 98 v4.77
Paint Shop Pro with Animation Shop 6.0
Reptile 2.0
SureThing CD Labeler 2.0
ThumbsPlus 4.02
Ulead WebRazor Pro 2.0
Xara3D 3.04

INTERNET
Añadir Pro 4.00.008
AutoWinNet 6.0 Beta 1
Copernic 99 v3.02
Cuentapasos 3.75
CuteFTP 3.0
Dial-Up Magic 1.8
Eudora Light 3.0.6
GetRight 4.0
Go!Zilla 3.5
Guardián 1.1
HomeSite 4.01
ICQ 99b beta 3.19 build 2569
MIRC 32 5.61
Net Vampire 3.3
PGP 6.0.2i
URL Organizer 2.3
WebTrends Enterprise Suite 3.6c
WinGate 3.0.5

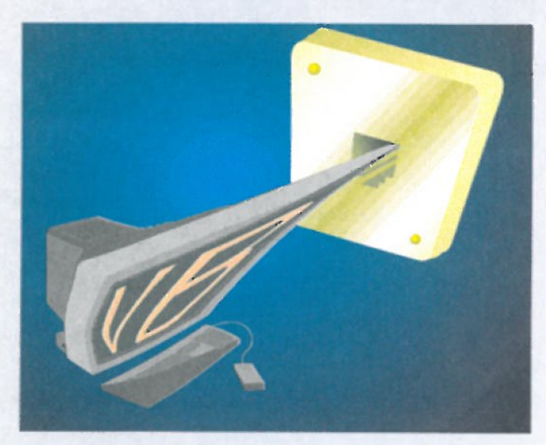
MULTIMEDIA
AudioCatalyst 2.01
COWON Jet-Audio 4.6
CDH Media Wizard 4.05
Sonique 1.05
WinAMP 2.50c

NAVEGADORES
Internet Explorer 5.0.2314
Netscape Communicator 4.61
Opera 3.60

PROGRAMACIÓN
Hackman 3.21
Help & Manual 2.22
InstallConstruct 3.2
Java Development Kit 1.2.2
UltraEdit Professional Text/HEX Editor 6.20b
Windows Registry Guide 1.3
WinHex 8.65

UTILIDADES
3DMark 99 Max b200
Adobe AcrobatReader 4.0
Advanced Registry Tracer (RegFix) 1.0a
Babylon Translator 2.1
CDRWin 3.7e
Emergency Recovery System 8.71
Nero 4.0.5.0
SiSoft Sandra 99 v5.10
Windows Commander 4.01
WinZip 7.0

ATENCIÓN:
En caso de problemas con el CD-ROM envíelo por correo ordinario, a la atención del SERVICIO TÉCNICO DE SÓLO P., incluyendo en el interior del sobre sus datos personales, a la siguiente dirección:
C/ San Sotero, Nº 5, 1ª Planta,
28037 (Madrid)



TAPI: API DE TELEFONÍA (I)

Constantino Sánchez Ballesteros. Técnico Superior de Desarrollo de Aplicaciones

Esta nueva serie de artículos tratará la programación de aplicaciones de telefonía bajo la *API* de *Windows*, *TAPI*. Con ella podremos controlar todos los aspectos referentes a la gestión de la línea telefónica.

¿QUÉ ES TAPI?

TAPI comprende la *API* de telefonía, dicho de otro modo, la Interfaz para la Programación de Aplicaciones de Telefonía. Fue desarrollada de forma conjunta por *Microsoft* e *Intel*, además de un gran número de compañías telefónicas y se realizó en 1994. *TAPI* permite a las aplicaciones *Windows* compartir dispositivos de telefonía unos con otros y provee una forma común para gestionar los diferentes medios (voz, datos, fax, vídeo, etc.) abarcando un amplio rango de plataformas *hardware*.

PLATAFORMAS COMPATIBLES CON TAPI

API está disponible para todas las plataformas *Windows* (desde la versión *3.11* hasta *NT*), con diferentes versiones de la interfaz para las diferentes versiones de *Windows*. La primera versión disponible fue la 1.3, una especie de *plug-in* para *Windows* 3.1. Esta versión ya no está soportada, aunque todavía podemos encontrar los ficheros y el programa de instalación en la página de desarrollo *MSDN* de *Microsoft*.

TAPI se encarga de la gestión de líneas telefónicas y hardware compatible

Windows 95 fue la primera versión de este sistema operativo que integró *TAPI* de manera directa. Cuando se realizó, *Windows* 95 soportaba la versión 1.4, una pequeña actualización de la anterior. Su característica más notable fue la habilidad de escribir aplicaciones de *TAPI* en 32 *bits*.

TAPI 2.0 se introdujo junto con la salida al mercado de *Windows NT* 4. No sólo fue la primera versión de *TAPI* soportada por *NT*,

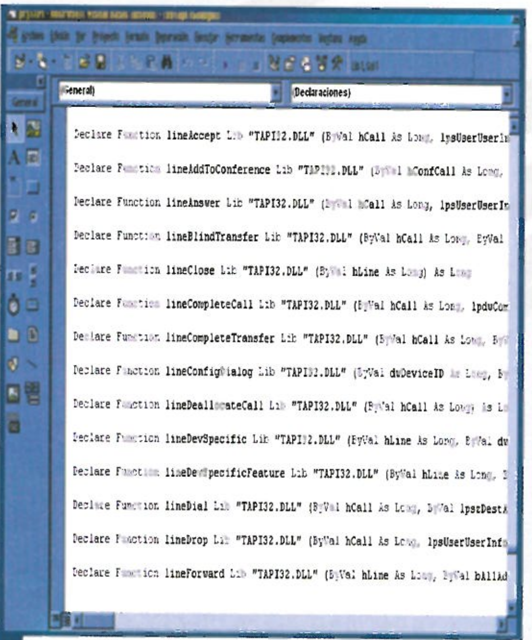


Figura 1.- En la parte negrita de la imagen sevisualizará la ayuda de *TAPI* dentro del entorno *Visual C++*.



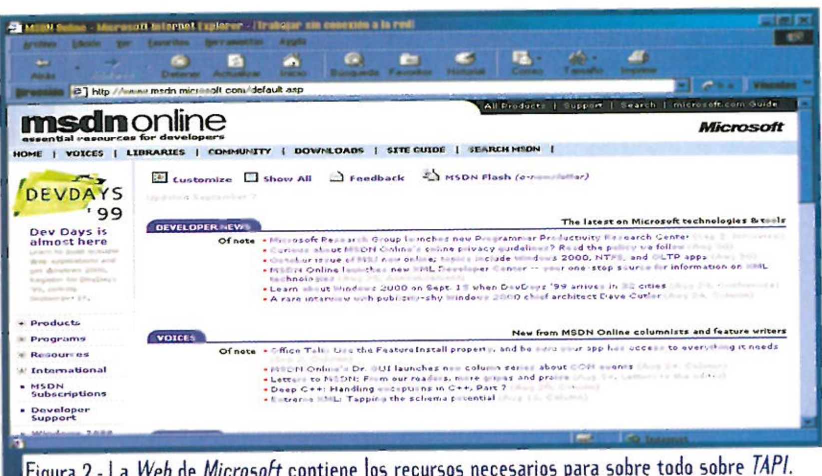


Figura 2.- La *Web* de *Microsoft* contiene los recursos necesarios para sobre todo sobre *TAPI*.

sino que además incluía un buen número de características nuevas como la inclusión del soporte *ACD* y otras características específicas de *PBX*.

A mediados del 97, *Microsoft* realizó *TAPI* 2.1, la primera versión que soportaba tanto por *Windows* 95 como *NT 4*. Actualmente *TAPI* 3.0 es la última revisión en la que ha trabajado *Microsoft* y es posible descargarla de la página *Web* de la empresa.

¿QUÉ ES UN TSP?

El término proveedor de servicios o *TSP* no es más que un nombre distintivo para un *driver*. Un *TSP* es un *driver* que permite a las aplicaciones *TAPI* comunicarse con diferentes tipos de *hardware TAPI*. En el caso de *Windows* 95 y *NT* conviene indicar que incluyen un *TSP* propio denominado *UniMódem*.

Se trata de un proveedor de servicios de Módem "universal" que soporta un amplio rango de módems utilizados comúnmente. Cuando utilizamos *hardware* de telefonía de diferente categoría al módem, como *PBX's*, tarjetas de procesamiento de voz, etc., utilizaremos un *TSP* provisto por el vendedor de ese tipo de *hardware* específico.

RELACIÓN ENTRE UN TSP Y EL HARDWARE TAPI

Un *TSP* traduce las funciones de *TAPI* en comandos que el *hardware* puede comprender, y a su vez traduce eventos del *hardware* en datos que la aplicación *TAPI* pueda entender. Dado que las diferentes clases de *hardware* de telefonía pueden soportar diferentes características, multitud de *TSP* soportarán diferentes funciones *TAPI*.

El TSP es el proveedor de servicios de TAPI

Es posible que un simple TSP que soporte diferentes tipos de hardware pueda tener un comportamiento distinto dependiendo del hardware que se esté utilizando. Un ejemplo de este comportamiento se puede encontrar en el amplio abanico de módems soportados por Windows.

Si el *TSP* que utilizamos soporta *CallerID* (identificador de llamada), y el módem que tenemos conectado también lo soporta, entonces una aplicación *Windows* sería capaz de obtener esa información vía *TAPI*. En el otro extremo, si utilizamos el mismo *TSP* con un módem que no soporte *CallerID* la aplicación no será capaz de obtener la información de la persona que llama.

ARQUITECTURA DE TAPI

Como introducción podemos relatar algunas de las posibles aplicaciones que pueden crearse con *TAPI*:

- Buzones de voz.
- Conferencia *IP*.
- Llamadas de voz sobre *Internet* utilizando el protocolo *H.323*.
- Llamadas básicas de voz a través de la red pública de telefonía (*PSTN*).
- Controles de *PBX*.
- Sistemas de respuestas de voz interactiva (*IVR*).

El siguiente diagrama ilustra la arquitectura básica de *TAPI*. Hay que reseñar que el rango de potencial de *TAPI 3.0* no se limita a utilizar *PSTN*, *ISDN* o transporte *TCP/IP*.

CREACIÓN DE APLICACIONES QUE SOPORTEN TELEFONÍA

Las características de la telefonía nos ayudan a obtener lo máximo de los sistemas de telecomunica-

ciones, permitiéndonos una gestión más eficiente de las llamadas de voz y el control de nuestras operaciones de transferencia de datos.

Podemos utilizar *TAPI* para implementar esta tecnología en bases de datos, procesadores de texto y cualquier aplicación que pueda beneficiarse del envío y recepción de datos a través de la línea telefónica.

TAPI nos ofrece una serie de herramientas para incorporar estas características en nuestras aplicaciones, como son:

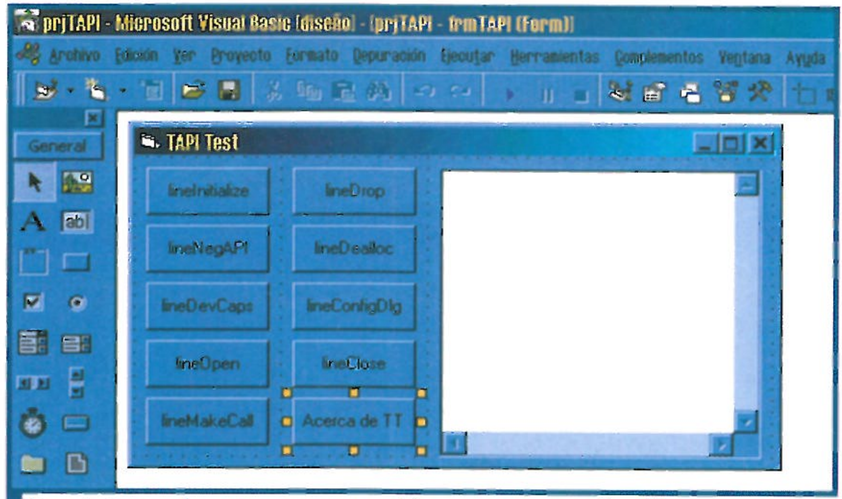


Figura 3.- Este es el aspecto externo que tendrá nuestro proyecto en *Visual Basic*.

- Conectar directamente a la línea telefónica en lugar de confiar en una aplicación individual de comunicaciones.
- Marcar números de teléfono automáticamente.
- Transmitir documentos como archivos, faxes o e-mails.
- Activar y gestionar las llamadas de las conferencias.
- Recibir, guardar y ordenar buzones de voz.
- Utilizar el *Caller-ID* para automatizar la gestión de llamadas entrantes.
- Controlar las operaciones de un ordenador remoto.

TAPI brinda el acceso en nuestra aplicación a la línea telefónica, al igual que nosotros proveemos a los usuarios el acceso a esa característica. Esto significa que podemos elegir y crear una interfaz de usuario que esté en consonancia con el resto de la aplicación. Por ejemplo, si utilizamos "arrastrar y soltar" (*drag and drop*) frecuentemente, podríamos permitir al usuario enviar archivos o faxes a través del teléfono soltando el icono del archivo sobre un icono que represente el destino al que se quiera enviar.

De forma similar, el usuario podría iniciar una conferencia soltando tres o cuatro nombres de una agenda electrónica en una "central de conferencias" y pulsar seguidamente el botón "conectar". El programador elige la interfaz y *TAPI* se encarga de realizar y gestionar las conexiones telefónicas, así de simple.

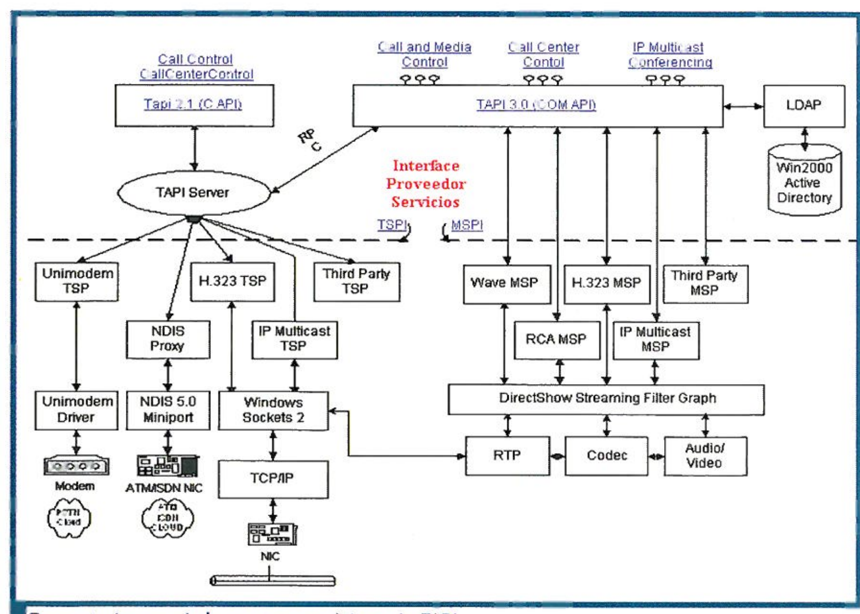


Figura 4.- Imagen de la arquitectura básica de *TAPI*.

SERVICIOS DE LA RED TELEFÓNICA

TAPI permite el acceso a través de una amplia variedad de servicios de la Red telefónica. Aunque estos servicios puedan utilizar diferentes tecnologías para establecer llamadas o transmitir voz y datos, *TAPI* se encarga de que estos detalles específicos para cada servicio sean transparentes de cara a la aplicación. Esto significa que podemos crear aplicaciones que puedan sacar partido de cualquier servicio disponible sin incluir código específico en nuestra aplicación.



Históricamente, muchas conexiones telefónicas del mundo han sido del tipo *POTS* (telefonía antigua). Muchas llamadas de *POTS* eran transmitidas digitalmente excepto el tramo llamado local *loop*, la parte de la línea telefónica que se encontraba entre el teléfono y la centralita de la compañía. Dentro de este bucle, la voz se transmitía en formato analógico y los datos digitales de un ordenador primero debían ser convertidos a analógicos por el módem. Actualmente, las redes digitales están reemplazando a las analógicas en el local loop.

Con TAPI podemos detectar la llamada entrante y su identificador

La utilización de *TAPI* sobre *POTS* es buena, puesto que es simple. Normalmente sólo se utiliza un tipo de información (como datos o voz) por llamada, soporta un canal por línea, etc. La amplia mayoría de usos que podemos dar a *TAPI* utilizan aún *POTS*, y muchos desarrolladores de telefonía utilizarán *TAPI* sólo con destino a aplicaciones que se basen en *POTS*.

Pero *TAPI* no sólo se restringe a los *POTS*, sino que permite realizar conexiones sobre otros tipos de redes. Actualmente se están desarrollando clases de transmisiones de datos más avanzadas. Por ejemplo, un importante servicio digital es la *RDSI* (Red Digital de Servicios Integrados), de la que se espera que aumente significantemente en eficacia. Las redes *RDSI* tienen las siguientes ventajas sobre los *POTS*:

- Toda la infraestructura es digital.
- Ofrece un menor índice de error.
- La transmisión de datos es más rápida, con velocidades que alcanzan los 128 *Kb* por segundo (*Kbps*) en servicio básico.
- De 3 a 32 canales para transmisiones simultáneas de voz y datos.
- Se trata de un estándar internacional.

Los ratios de error de la *RDSI* son más bajos que la transmisión analógica porque los datos viajan de un punto de la *RDSI* a otro en formato digital. El aumento de velocidad a 128 *Kbps* es posible sobre líneas básicas y mucho mayor sobre líneas primarias. Cuando las conexiones sobre *RDSI* se extiendan, los usuarios serán capaces de enviar datos de forma simultánea con llamadas de voz a otra persona. Cada línea *RDSI*, dependiendo de su ratio de transmisión, provee al menos tres canales (dos para voz o datos y uno estrictamente para datos o información exclusiva) llegando hasta 32 canales, para gestión simultánea de datos y voz.

Las líneas básicas permiten dos canales de 64 *Kbps* tipo B (para enviar voz o datos) y uno de 16 *Kbps* tipo D (para información exclusiva o datos empaquetados). Las líneas primarias americanas y japonesas tienen 23 canales tipo B de 64 *Kbps* y uno tipo D de 64 *Kbps*. Las europeas ofrecen 30 canales tipo B y dos canales de tipo D.

TAPI puede utilizarse tanto con líneas analógicas como digitales

TAPI también se puede utilizar con otras redes digitales como la T1/E1.

Figura 5.- Detección de dispositivos instalados

COMPONENTES

Basándose en el modelo de arquitectura de servicios abierta de *Windows* (*WOSA*), la telefonía bajo *Windows* consiste en TAPI y las *DLL* de *TAPI32*, *tapisrv.exe* (que implementa y gestiona las funciones *TAPI*) y uno o más *TSP*. *TAPI* se define como una interfaz independiente del dispositivo para llevar a cabo las tareas de telefonía.

Los proveedores de servicios son *DLLs* que gestionan el *hard-*

Listado 1: Creación de un objeto TAPI con C++ y Visual Basic.

```
Código en C++:
// Inicializamos COM
CoInitializeEx(NULL, COINIT_MULTITHREADED );

// Creamos un punto de entrada al objeto TAPI
CoCreateInstance(CLSID_TAPI, NULL,
CLSCTX_INPROC_SERVER, IID_ITTAPI,
(LPVOID *)&gpTapi );
// Inicializamos TAPI
gpTapi->Initialize();

Código en Visual Basic
'Declaramos objeto TAPI de forma global
Dim gobjTapi As TAPI
Dim glRegistrationToken As Long
Dim gobjReceivedCallInfo As ITCallInfo
Dim gbSupportedCall As Boolean

'Creamos el objeto TAPI
Set gobjTapi = New TAPI

'Llamamos a Initialize antes de
'hacerlo a cualquier otra función de TAPI
Call gobjTapi.Initialize
```

```
(General)        (Declaraciones)
Declare Function lineAccept Lib "TAPI32.DLL" (ByVal hCall As Long, lpsUserUserIn
Declare Function lineAddToConference Lib "TAPI32.DLL" (ByVal hConfCall As Long,
Declare Function lineAnswer Lib "TAPI32.DLL" (ByVal hCall As Long, lpsUserUserIn
Declare Function lineBlindTransfer Lib "TAPI32.DLL" (ByVal hCall As Long, ByVal
Declare Function lineClose Lib "TAPI32.DLL" (ByVal hLine As Long) As Long
Declare Function lineCompleteCall Lib "TAPI32.DLL" (ByVal hCall As Long, lpdwCom
Declare Function lineCompleteTransfer Lib "TAPI32.DLL" (ByVal hCall As Long, ByV
Declare Function lineConfigDialog Lib "TAPI32.DLL" (ByVal dwDeviceID As Long, By
Declare Function lineDeallocateCall Lib "TAPI32.DLL" (ByVal hCall As Long) As Lo
Declare Function lineDevSpecific Lib "TAPI32.DLL" (ByVal hLine As Long, ByVal dw
Declare Function lineDevSpecificFeature Lib "TAPI32.DLL" (ByVal hLine As Long, B
Declare Function lineDial Lib "TAPI32.DLL" (ByVal hCall As Long, ByVal lpszDestA
Declare Function lineDrop Lib "TAPI32.DLL" (ByVal hCall As Long, lpsUserUserInfo
Declare Function lineForward Lib "TAPI32.DLL" (ByVal hLine As Long, ByVal bAllAd
```

Figura 6.- Declaración de funciones en *Visual Basic* para *TAPI*.

ware a bajo nivel y posibilitan las acciones necesarias para completar las tareas de telefonía a través de los dispositivos *hardware* (faxes, tarjetas *RDSI*, teléfonos y módems). Las aplicaciones sólo enlazan y llaman a las funciones de la *DLL* de *TAPI*; nunca llaman a los proveedores de servicios de forma directa.

Cuando una aplicación llama a una función de *TAPI*, la *DLL* valida y ejecuta los parámetros de la función enviándolos a *tapisrv.exe*. *TAPISRV* (el servicio de telefonía) procesa la llamada y envía una petición al proveedor de servicios apropiado. Para recibir peticiones de *TAPISRV*, el proveedor de servicios debe implementar una interfaz de proveedor de servicios de telefonía (*TSPI*). Un proveedor de servicios puede proveer diferentes niveles de la interfaz:

- Básico.
- Suplementario.
- Extendido.

Por ejemplo, un simple proveedor de servicios puede facilitar servicios de telefonía básica, como soporte para llamadas salientes a través de un módem compatible *Hayes*. Un proveedor de servicios particular, desarrollado por una empresa cualquiera, puede proveer un amplio soporte de llamadas entrantes y salientes.

Un usuario puede instalar cualquier número de proveedores de servicios sobre un ordenador siempre que estos proveedores no intenten acceder al mismo dispositivo hardware al mismo tiempo. El usuario asocia el *hardware* y el proveedor de servicios cuando se instalan. Existen algunos proveedores capaces de acceder a múltiples dispositivos. En algunos casos, el usuario puede que necesite instalar un driver especial



Listado 2: Programación de una apertura de línea con Visual Basic

```
Private Sub Command5_Click()
'
' Apertura de línea
'
Dim a As Long
'
udtLineCall.dwTotalSize = LINECALLPARAMS_FIXEDSIZE + 2048

a = lineOpen(lineApp, 0, lphLine, 65540, &H0,
    AddressOf LINECALLBACK,
    LINECALLPRIVILEGE_NONE,
    LINEMEDIAMODE_DATAMÓDEM,
    udtLineCall)

If a <> 0 Then
 AddText Me.Text1, TapiErrMsg(a) & " - lineOpen"
Else
 AddText Me.Text1,
 Hex(udtLineCall.dwMediaMode) & " - lineOpen"
End If

End Sub
```

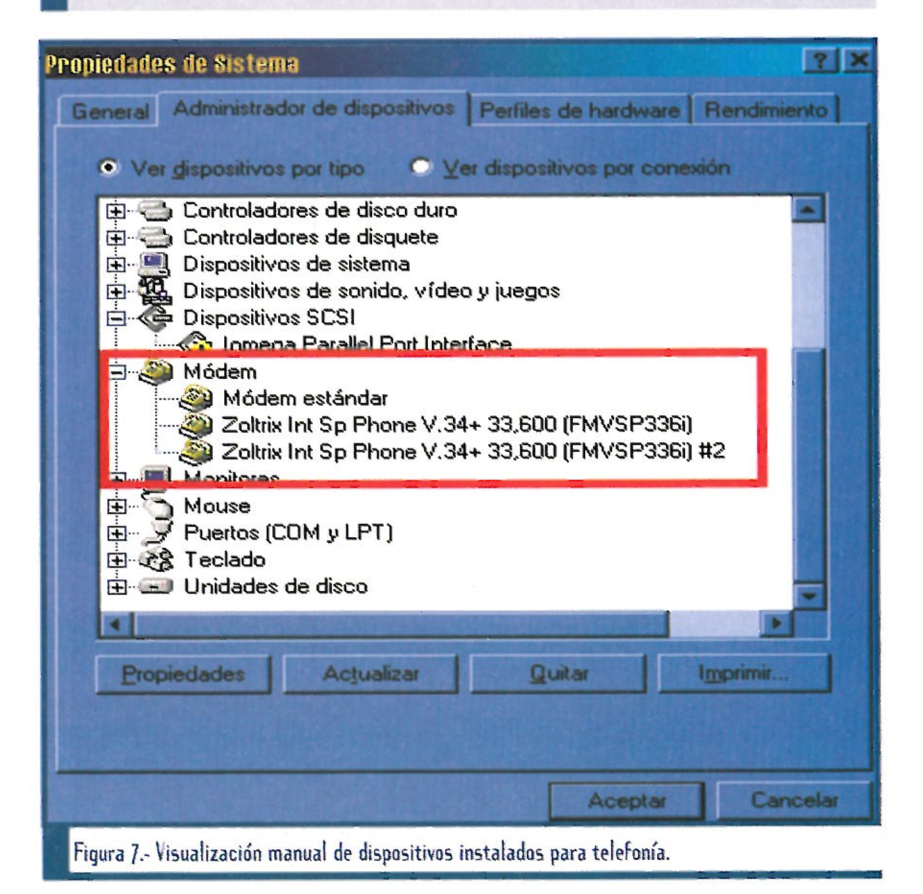


Figura 7.- Visualización manual de dispositivos instalados para telefonía.

junto con el proveedor de servicios.

Nuestras aplicaciones pueden utilizar las funciones *TAPI* para determinar los servicios que están disponibles en el ordenador. *TAPI* determina qué proveedores de servicios están disponibles y devuelve información sobre sus características a la aplicación. De este modo, cualquier número de aplicaciones puede requerir servicios del mismo proveedor, ya que *TAPI* gestionará todos los accesos.

MEDIA STREAM

Consiste en la información intercambiada sobre una llamada. *TAPI*, de forma autónoma, provee control sólo para la línea y los dispositivos telefónicos y no permite acceso al contenido del media stream. Para manejarlo, una aplicación debe utilizar *API's* de *Win32*, como la de comunicaciones, Wave Audio, o *MCI*. Por ejemplo, una aplicación que provea una interfaz para el manejo de un fax o transmisión de datos utiliza funciones *TAPI* para controlar y monitorizar la línea sobre la que se envían los bits, pero utiliza las *API's* de comunicaciones para transmitir los datos.

Media Stream en llamadas de voz es controlado por TAPI, por el usuario

Del mismo modo, media stream en una llamada de voz no es controlado y producido por *TAPI*, sino por una persona que habla a la otra. De cualquier modo, la línea sobre la que la llamada es establecida y monitorizada, además de la propia llamada, perma-

Listado 3: Ejemplo de utilización de la función LineClose.

```
Private Sub cmdClose_Click()
'
' Cierre de línea
'
Dim retcode As Long
'
retcode = lineClose(lphLine)

If retcode <> 0 Then
 AddText Text1,
 TapiErrMsg(retcode) & " - lineClose"
Else
 AddText Text1, "Línea cerrada! - lineClose"
End If
'
End Sub
```

necen en control de la aplicación *TAPI*. Hay que reseñar que la voz es considerada como una señal que puede viajar sobre un canal con un ancho de banda de 3.1 *KHertzios*.

HARDWARE EXCLUSIVO

Algunas de las características más avanzadas de *TAPI* requieren que una aplicación sea capaz, por ejemplo, de obtener datos del teléfono. Muchos teléfonos no pueden estar conectados directamente a ordenadores para controlar las llamadas de voz y de este modo, son incapaces de soportar funciones de telefonía.



Figura 8.- El control de comunicaciones se puede combinar con *TAPI* para implementar funciones.

C++ Vs. VISUAL BASIC

Podemos utilizar *TAPI* tanto con *C++* como con *Visual Basic*. Por supuesto, en *C++* tendremos más posibilidades a la hora de programar que las ofrecidas por *Visual Basic*.

Llegado el momento de codificar, las cosas no varían demasiado de un lenguaje a otro, siempre refiriéndonos a la metodología de trabajo en relación a las funciones utilizadas. Prueba de ello es el extracto de código, que se muestra en listado 1 el cual se encarga de crear un objeto *TAPI* y seleccionar una dirección apropiada.

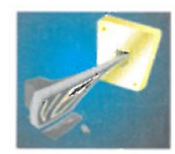
APERTURA DE LÍNEAS

Para empezar a utilizar una línea telefónica mediante *TAPI* el primer paso es obtener sus características. Después, la aplicación debe abrir el dispositivo de línea antes de acceder a las funciones de telefonía. Cuando un dispositivo de línea ha sido abierto con éxito, la aplicación recibe un *handle* del mismo. La aplicación podrá entonces utilizar esa línea para gestionar las llamadas entrantes, crear las llamadas salientes o monitorizar las actividades de la llamada establecida en la línea.

La función encargada de abrir un dispositivo de línea *LineOpen*. Posteriormente, cuando tengamos que cerrar este dispositivo utilizaremos la función *LineClose*. La función *lineOpen* puede invocarse de dos modos:

- Especificando el dispositivo de línea: puede seleccionarse mediante su identificador (parámetro *dwDeviceID*). La petición de *lineOpen* abrirá el dispositivo de línea especificado. Las aplicaciones interesadas en gestionar las llamadas entrantes utilizan dispositivos de línea específicos porque la aplicación ha sido notificada sobre qué línea se está utilizando o espera a la llamada entrante. Cuando un dispositivo de línea se ha abierto con éxito, la aplicación retorna un *handle* representando la línea abierta.

- Especificando las propiedades deseadas para cualquier dispositivo de línea. La aplicación puede especificar que quiere utilizar cualquier dispositivo de línea que tenga ciertas propiedades. En este caso,



```
┌── lineInitialize();              // Inicializa TAPI para usar el dispositivo de línea
│
│     lineNegotiateAPIVersion();   // Negocia número de versión de la API
│  ┌── lineOpen();                 // Abre el dispositivo de línea
│  │
│  │  ┌── lineMakeCall();          // Colocamos una petición de llamada asíncrona
│  │  │
│  │  │     // Llamada
│  │  │     // Transferencia de datos
│  │  │     // Transferencia de Fax
│  │  │     // etc.
│  │  │
│  │  └── lineDrop();              // Termina la llamada
│  │
│  └── lineClose();                // Libera el dispositivo de línea
│
└── lineShutdown();                // Termina TAPI
```

Figura 9.- A modo de resumen, estos son los pasos a seguir para crear una aplicación en *TAPI*.

la aplicación utilizará el valor *LINEMAPPER* en lugar de un identificador específico para el parámetro *dwDeviceID*. La aplicación también especifica qué propiedades se necesitan en la llamada utilizando los parámetros de *lineOpen*. La función abre cualquier dispositivo de línea disponible que soporte los parámetros de llamada especificados. Por supuesto, este intento puede fallar. Si se logra, el que llama puede determinar el identificador de dispositivo de línea mediante la función *LineGetID*, especificando el handle (*lph Line*) del dispositivo abierto retornado por *lineOpen*.

Una aplicación que haya abierto un dispositivo de línea puede utilizarlo para realizar una llamada saliente excepto si la línea sólo soporta llamadas entrantes.

En el listado 2 podemos comprobar la programación de una apertura de línea mediante *Visual Basic*. Para cerrar una línea utilizaremos la función *LineClose*. En el Listado 3 podemos apreciar un extracto de código realizando esta tarea.

SELECCIÓN DE UNA O MÁS LÍNEAS

Una aplicación puede abrir una o más líneas para varios propósitos. Por ejemplo, puede abrir una línea para monitorizar llamadas y otra línea para crear llamadas salientes. Si hay disponibles varias líneas, la aplicación puede elegir la apertura de cualquiera de ellas o todas. Para decidir qué dispositivos de línea utilizar, se deben determinar las características de cada una mediante la función *LineGetDevCaps*.

En el siguiente extracto de código se utiliza *LineGetDevCaps* para obtener características de la línea:

```
Dim a As Long
Dim b As LINEDEVCAPS

b.dwTotalSize =
    LINEDEVCAPS_FIXEDSIZE + 2048
a = lineGetDevCaps(lineApp, x,
    65540, &H0, b)
```

Esta función nos dice si la línea soporta la funcionalidad requerida por las llamadas entrantes o salientes, así como su *media mode* requerido. Esta función también se utiliza para obtener el nombre.

Dado que la apertura de una línea significa obtener un *handle* a la misma (*hLine*) con un privilegio determinado, una aplicación puede obtener más de un *handle* para la misma línea. En otras palabras, una aplicación puede abrir la misma línea varias veces. Este hecho se conoce como la apertura de diferentes instancias de la línea.

Por ejemplo, una línea puede ser abierta simultáneamente una vez para monitorización de llamadas, una segunda para aceptar llamadas entrantes y una tercera para crear llamadas salientes.

Después de elegir una línea determinada (o líneas), la aplicación utilizará *LineOpen*, especificando una línea determinada o utilizando el parámetro *LINEMAPPER*, tal y como se explicó en un apartado anterior.

LIBRERÍAS

Para poder compilar con éxito la mayoría de programas *TAPI* 3.0 que se creen bajo *C++*, será necesario incluir las siguientes librerías:

- advapi32.lib
- amstrmid.lib (ActiveMovie™ GUIDs)
- kernel32.lib
- mdhcpid.lib (multicast GUIDs)
- ole32.lib COM
- oleaut32.lib (COM Automation)
- rendid.lib (Rendezvous GUIDs)
- rpcndr.lib
- rpcns4.lib
- rpcrt4.lib
- sdpblbid.lib (Session Descriptor Protocol (SDP) GUIDs)
- strmiids.lib
- t3iid.lib (TAPI 3.0 GUIDs)
- user32.lib
- uuid.lib
- wldap32.lib
- ws2_32.lib

Si utilizamos *Microsoft Visual Studio*, es posible que necesitemos actualizar nuestra versión. En particular, el archivo *LINK.EXE* debe haber sido creado el 19/3/98 ó fecha superior.

Las sentencias *#define* del compilador deberán incluir los siguientes parámetros:

```
_WIN32_WINNT 0x500
 _WIN32_DCOM.
```

PRECAUCIONES AL PROGRAMAR BAJO C++

Cuando programemos *TAPI* 3.0 bajo *C++* debemos poner atención a los siguientes detalles:

La función *COM CoInitialize* crea de forma indirecta una ventana (esto es especialmente importante en el apartado multitarea). Si una tarea crea una ventana, ésta debe procesar mensajes. Por ello, si se llama a *CoInitialize*, necesitaremos ejecutar un bombeo de mensajes para prevenir los problemas que se deriven en la ventana. Si no es así, métodos de interfaces COM como *IGlobalInterface*Table podrían colgarse.

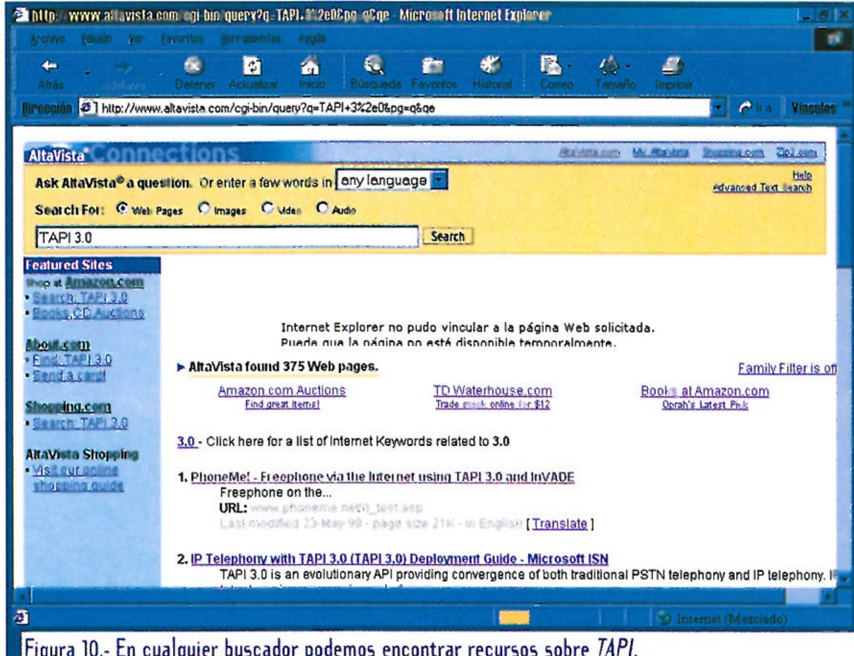


Figura 10.- En cualquier buscador podemos encontrar recursos sobre *TAPI*.

En el apartado de "hilos" (*threading*) debemos tener un bombeo de mensajes activo sin tener en cuenta si estamos esperando sincronización de objetos.

Debemos tener cuidado cuando llamemos a funciones de sincronización o espera (*Sleep()*), *WaitForMultipleObjects*, *WaitForMultipleObjectsEx*, *WaitForSingleObjectEx*, etc., en su lugar utilizaremos *MsgWaitForMultipleObjects* y procesaremos los mensajes, o utilizaremos *CoWaitForMultipleHandles*, que automáticamente detectará en qué apartado se encuentra el hilo (*STA* o *MTA*, es decir, tarea simple o multitarea, respectivamente) y esperará tanto si utilizamos un *loop* modal de *COM* como si lo hacemos mediante *WaitForMultipleObjects*.

MsgWaitForMultipleObjects y *CoWaitForMultipleHandles* también procesan mensajes de forma acorde a las reglas de *COM*. Por ejemplo, en lugar de:

```
Sleep (5000);
```

Deberíamos utilizar:

```
{
DWORD dwSignalled;

HANDLE heventDone =
    CreateEvent(0, FALSE, FALSE,
    0);

CoWaitForMultipleHandles
    (COWAIT_ALERTABLE, 5000, 1,
    &heventDone, &dwSignalled);

CloseHandle(heventDone);
}
```

Este sistema de trabajo es algo más complicado pero más eficiente a la hora de ejecutar la aplicación.

En el siguiente artículo abordaremos la creación de un proyecto en *Visual Basic* que haga llamadas a la *DLL TAPI32* para utilizar las funciones más comunes relacionadas con la telefonía bajo *TAPI*.

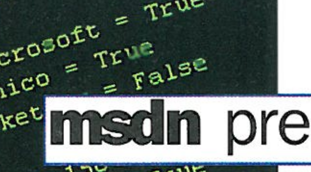
Más de 750.000 páginas de documentación • 40.000 artículos técnicos • Truco y técnicas de programación • 1.500 ejemplos de código documentado para reutilizar en sus aplicaciones • Autoformación y seminario OnLine sobre las últimas tecnologías: XML, DHTML, SOAP, Windows DNA 2000, Windows 2000, VBA • El calendario más completo de eventos técnicos para desarrolladores • Libros técnicos y revistas OnLine para programadores • Áreas de descarga de software • Completa información sobre herramientas de desarrollo.

Toda esta información la encontrarás en el nuevo portal en castellano para desarrolladores, programadores y analistas

msdn.microsoft.es

y además toda la información sobre MSDN Presentaciones





La información clave para la comunidad de desarrolladores

¡Infórmate ya, plazas limitadas!

msdn
Microsoft Developer Network

msdn presentaciones



En una mañana trataremos en detalle los temas técnicos de mayor actualidad. Son eventos gratuitos, impartidos por los mejores Ingenieros de Microsoft, exclusivamente para técnicos como tú. Además contaremos con la presencia de Compaq, que contará su experiencia en la implantación de un caso real y de los centros de formación Microsoft CTEC Professional Training e Intec.
No puedes perderte este evento clave para el desarrollador.

MADRID

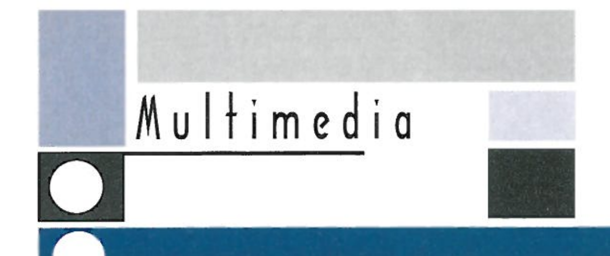
FECHA	CÓDIGO	ÁREAS
11 noviembre	MSDN00P5	Migración Windows 2000
23 noviembre	MSDN00P8	Datawarehousing
16 diciembre	MSDN00P10	Gestión de conocimiento

BARCELONA

FECHA	CÓDIGO	ÁREAS
16 noviembre	MSDN00P6	Migración Windows 2000
18 noviembre	MSDN00P7	Datawarehousing
14 diciembre	MSDN00P9	Gestión de conocimiento

Para información detallada de la agenda e inscripción al evento, consulta
msdn.microsoft.es/eventos/
o llama a nuestro teléfono de atención al cliente **902 197 198**







Programar un reproductor MM con Delphi (I)

Juan Luis Ceada Ramos
Programador en ARCABE Formación y Servicios Informáticos.

En este número comenzamos una nueva serie de artículos durante los cuales veremos cómo desarrollar un reproductor multimedia usando como lenguaje de programación Delphi.

A lo largo de esta serie veremos lo sencillo que resulta crear un reproductor multimedia usando las herramientas que *Inprise* (antes *Borland*) y *Microsoft* nos proporcionan.

Nuestro reproductor trabajará sin problemas con ficheros de sonido y vídeo. Incluso, si nos lo proponemos, podría controlar un vídeo, un *DVD*, un *CD* o un *laserdisc* (por ejemplo). Se darán algunas nociones de programación para aquellos que quieran conseguir controlar un dispositivo externo desde el reproductor, aunque durante esta serie sólo nos limitaremos a reproducir archivos de audio y vídeo.

La aplicación reproducirá múltiples formato de fichero

Existen multitud de formatos de audio y vídeo, de tal forma que nuestro reproductor soportará diferentes formatos en función del *software* que tengamos instalado en nuestra máquina. Más adelante abordaremos este aspecto con mayor detalle.

Además, aprovecharemos para responder a algunas de las preguntas que más suelen hacerse en los foros de programación. Por ejemplo, veremos la forma de hacer que nuestra aplicación acepte los ficheros que se arrastran sobre ella, la forma de apagar el ordenador mediante código, así como otros truquillos que nos pueden venir muy bien en otros proyectos de mayor envergadura.



¿QUÉ NECESITAMOS?

Podemos optar por dos formas de implementación. Una es la sencilla (y rápida), que consistiría en utilizar algunos de los componentes que existen en el mercado, y simplemente realizar las llamadas a los procedimientos oportunos. La otra es algo más complicada, y consiste en programar nosotros mismos todo lo relacionado con la reproducción de los diferentes ficheros, basándonos en el estándar MCI.

Este mes empezaremos por la parte fácil. Trabajaremos con el control *ActiveX TMediaPlayer* que

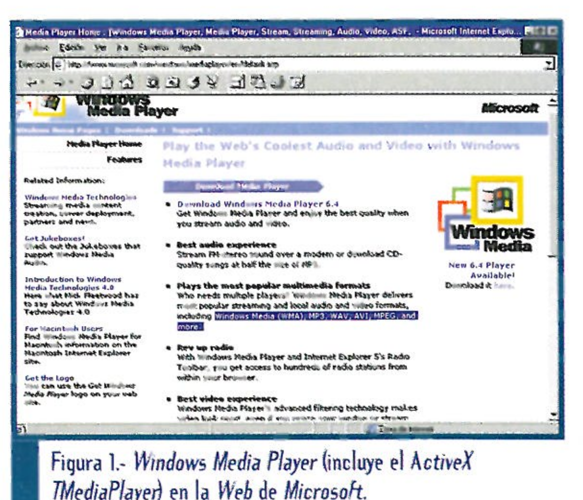


Figura 1.- *Windows Media Player* (incluye el *ActiveX TMediaPlayer*) en la *Web* de *Microsoft*.



proporciona *Microsoft* (se puede descargar de: http://www.microsoft.com/windows/mediaplayer/en/default.asp).

Este control es, desde mi punto de vista, superior al *TMediaPlayer* que se distribuye con *Delphi*. Destaca por una mayor facilidad de uso, combinado con una serie de características más amplias. Eso sí, al final con los dos componentes es posible realizar prácticamente las mismas aplicaciones (puesto que ambos basan su funcionamiento interno en la interfaz *MCI*).

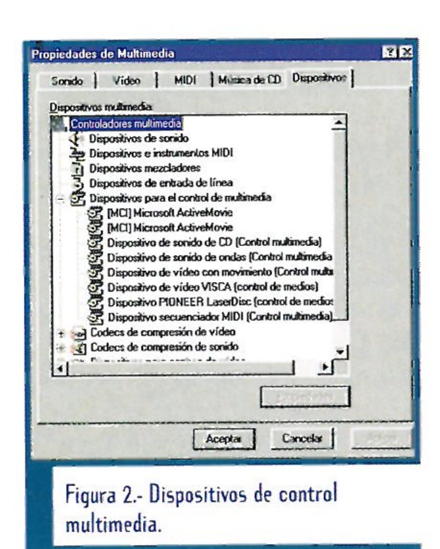


Figura 2.- Dispositivos de control multimedia.

¿QUÉ ES MCI?

Como hemos dicho, los controles de tipo *TMediaPlayer* basan su funcionamiento en la interfaz *MCI* (*Multimedia Control Interface*: Interfaz de Control Multimedia).

El estándar *MCI* proporciona una interfaz de alto nivel, que podrá ser utilizada por el programador para desarrollar sus aplicaciones multimedia, independientemente de los dispositivos y ficheros utilizados. En cualquier caso, también es posible controlar los dispositivos multimedia mediante funciones de bajo nivel, aunque resulta recomendable utilizar la interfaz *MCI*.

Usaremos el control ActiveX Media Player que nos proporciona Microsoft

Usando *MCI*, un programa tendrá la capacidad de reproducir y grabar en cualquier dispositivo multimedia soportado por el sistema. Esto incluye a los más usuales (ficheros *WAV*, *MIDI*, *CD-Audio*), y a otros que ya no lo son tanto (como vídeos, equipos de cintas *DAT*, etc.)

TIPOS DE ARCHIVOS

El *ActiveX TMediaPlayer*, al tener como base la interfaz *MCI*, permite controlar un número variable de dispositivos multimedia. Dicho número depende del *software* que tengamos instalado en nuestro ordenador. En la Figura 2 se muestran los controladores de dispositivos multimedia que se instalan con *Windows 98*. Como se puede apreciar, a través del reproductor multimedia, desde mi equipo puedo controlar un *CD-Audio*, un secuenciador *MIDI*, un *laserdisc*, así como ficheros de sonido y vídeo.

Como ya comentamos anteriormente, en este caso nos centraremos en la reproducción de vídeo y audio en movimiento. Los ficheros que se podrán reproducir dependen del *software* instalado y más concretamente, de los *codecs*.

¿QUÉ ES UN CODEC?

Un *codec* no es más que un controlador capaz de codificar/descodificar un determinado flujo de datos. Básicamente, cuando mediante *MCI* solicitamos reproducir un fichero de audio, se busca el *codec* que es capaz de reproducirlo. El *codec*, responde a las peticiones de la interfaz *MCI*, descodificando los datos, y procediendo a su reproducción. En la Figura 3 podéis ver los *codecs* de audio y vídeo existentes en la máquina utilizada para el desarrollo del artículo.

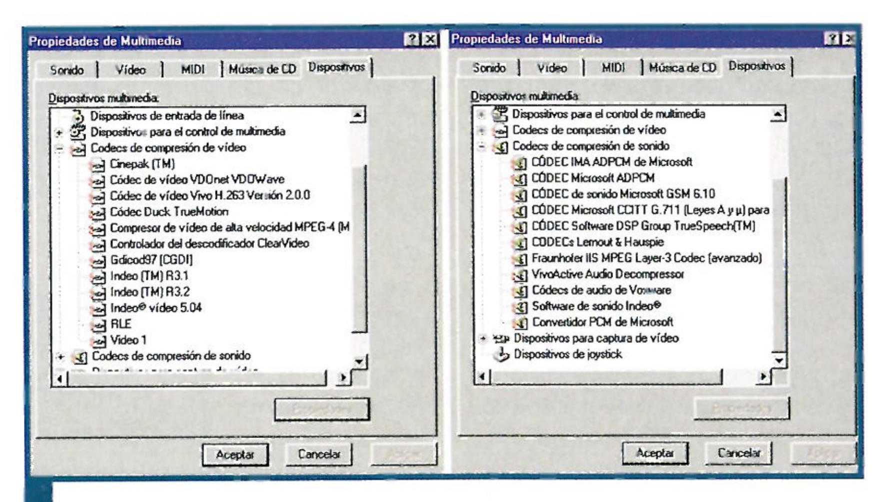


Figura 3.- *Codecs* de audio y vídeo instalados en un sistema con *Windows* 98.

En la Tabla 1 se pueden observar todos los tipos de ficheros que se pueden reproducir con estos *codecs*. Además, es posible añadir nuevos *codecs* en nuestra máquina, para poder reproducir, por ejemplo, ficheros *MOD* y *S3M* (ver: http://www.dc.ee/Files/Programm.Sound/)

Tabla 1. Tipos de ficheros que se pueden reproducir

Extensión	Descripción
asf, asx, wm	Audio
wma, wax	*Windows Media Audio*
wma, wvx	*Windows Media Audio & Video*
avi	Videos en formato *AVI*
wav	Audio en formato *WAV*
mpeg, mpg, mpe, m1v, mp2,mp2v, mpa	Videos comprimidos usando el estándar *MPEG*
mp3, m3u	Ficheros *MP3* (audio)
mid, midi, rmi	*MIDI* (audio)
Ivf	*Intel Indeo Video*
aif, aifc, aiff	Audio en formato *AIFF*
au, snd	Audio en formato *AU*
mov, qt	*QuickTime Videos*

IMPLEMENTACIÓN

Antes de empezar, recordemos que hay que descargar (e instalar) de la página *Web* de *Microsoft* el control *Windows Media Player.* Que viene incluido en el nuevo *IE* 5.0, aunque tampoco está de más pasar por la *Web* y buscar la última versión.

AÑADIENDO LOS COMPONENTES

En primer lugar debemos instalar el control *ActiveX TMediaPlayer* en *Delphi*. Para ello, seleccionaremos del menú principal *Component->Import ActiveX Control.* Buscaremos en la lista el control *Windows Media Player* (ver Figura 4), el cual instalaremos con un nombre de clase *TMMediaPlayer* (con dos "m", ya que si no entraría en conflicto con el componente *TMediaPlayer* que se distribuye con Delphi).

Por cierto, como se aprecia en la Figura 4, hay muchos controles *ActiveX* en *Windows* que podemos usar en nuestras aplicaciones, y que abarcan prácticamente todos los campos (manejo de imágenes, *Internet,* control de puertos serie, etc.). No estaría de más perder un poco de tiempo investigando los que hay instalados en nuestro sistema.

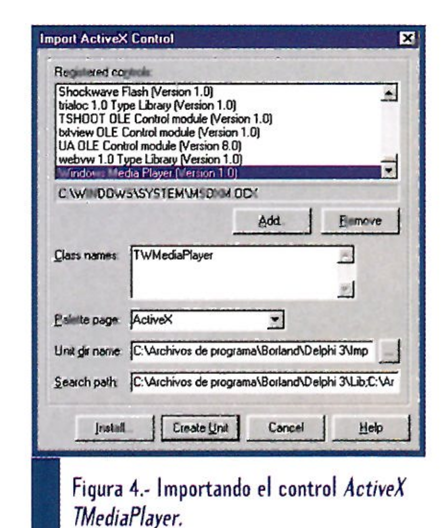


Figura 4.- Importando el control *ActiveX TMediaPlayer.*

COMPONENTE TMEDIAPLAYER

A continuación insertaremos el componente, al que llamaremos MP y asignaremos los valores adecuados a las propiedades que se muestran en la Tabla 2. Además, en las Tablas 3 y 4 se muestran los métodos y eventos más importantes de este componente.

En esta entrega sólo vamos a usar el método *Open* (fichero), y el evento *OnWarning.* Para el resto de funciones (como por ejemplo, avanzar en la reproducción, parar, pasar a la siguiente canción, etc.), dejaremos el control al propio componente, que es casi una pequeña aplicación por sí sólo.

El método *Open* abre y reproduce (si **AutoStart** igual a **True**) el fichero que le indiquemos. A diferencia de la propiedad *FileName,* ésta trabaja de forma asíncrona (es decir, el código continúa ejecutándose aunque el fichero aún no haya terminado de abrirse). En caso de que *AutoStart* sea igual a **False,** hay que llamar al método *Play* para que comience la reproducción comienzo.

Los codecs permiten reproducir diferentes formatos de archivo

En el evento *OnWarning* mostramos al usuario todos los errores que se produzcan durante la ejecución del programa que estén relacionados con este control.

En el Listado 1, incluido en el *CD-ROM* se muestra todo el código que va asociado a este control. El método *Open* se incluye dentro de una función auxiliar denominada *ReproducirFichero,* y que recibe como parámetro el nombre del fichero que se va a reproducir.

Es posible reproducir varios ficheros *MP3,* si abrimos un fichero con extensión *m3u.* El componente se encarga de interpretar qué ficheros contiene la lista, y los reproduce uno detrás de otro. Aquí es donde cobra sentido la opción de apagado automático, que veremos a continuación.

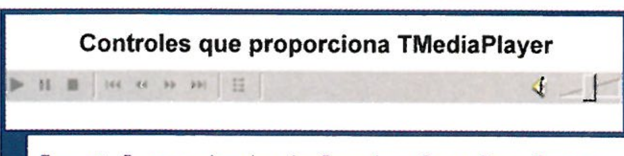


Figura 5.- De izquierda a derecha: Reproducir, Pausa, Parar, Primero, Anterior, Siguiente, Último, Previo, Silencio y Volumen.



Tabla 2. Principales propiedades del control ActiveX *TMediaPlayer*.

Propiedad	Valor	Descripción
AllowChangeDisplaySize	*True*	Permitir el cambio del tamaño de visualización
AutoRewind	*False*	Volver al punto inicial después de finalizar la reproducción
AutoSize	*False*	Permitir que el tamaño del control se reproducción
AutoStart	*True*	Empezar a reproducir en cuanto finalice la carga del fichero
EnableContextMenu	*False*	Mostrar un menú contextual al pulsar con el botón derecho
Filename	*False*	Nombre del fichero que se va a reproducir. Al rellenar esta propiedad, automáticamente se llama al método*Open* de forma bloqueante.
PlayState	*Sólo Lectura*	Indica el estado actual del reproductor. Posibles valores: **mpStopped, mpPaused, mpPlaying, mpWaiting, mpScanForward, mpScanReverse, mpClosed**
ReadyState	*Sólo Lectura*	Devuelve el estado actual del componente. Posibles valores: **mpReadyStateUninitialized, mpReadyStateLoading, MpReadyStateInteractive, MpReadyStateComplete**
Duration	*Sólo Lectura*	Indica la duración del archivo reproducido en segundos.

BOTONES

Los botones se encuentran dentro de un componente *TToolBar*. Además, existe un componente *TImageList* que contiene las imágenes que mostrarán dichos botones.

El primer botón muestra un cuadro de diálogo para seleccionar un fichero, y una vez seleccionado, lo reproduce.

Ahora sólo se muestran algunas posibilidades de TMediaPlayer

El segundo botón muestra un nuevo cuadro de diálogo donde el usuario puede introducir la hora (en formato *HH:MM*) a la que el ordenador se apagará automáticamente. Esto puede ser útil, por ejemplo, en caso de que nos guste irnos a la cama escuchando música, pero no nos apetezca dejar el ordenador encendido toda la noche.

Para saber si la hora actual coincide con la introducida se ha añadido



Tabla 3. Principalels métodos *TMediaPlayer.*

Método	Descripción
Open(fichero)	Abre el fichero indicado (y si *AutoStart*=**True,** comienza su reproducción).
Pause	Detiene momentáneamente la reproducción.
Stop	Detiene definitivamente la reproducción.
Next	Reproduce el siguiente fichero.
Previous	Reproduce el fichero anterior.
FastFoward	Avanza rápidamente (si el dispositivo lo soporta).
FastReverse	Retrocede rápidamente (si el dispositivo lo soporta).

Tabla 4. Principalels eventos *TMediaPlayer.*

Evento	Descripción
OnEndStream	Se produce cuando se ha finalizado la reproducción de un fichero.
OnPlayStateChange	Se produce cuando cambia el estado del reproductor (por ejemplo, de *Play a Stop*).
OnError	Se produce cuando hay un error durante la reproducción o carga.
OnWarning	Se produce cuando hay un error durante la reproducción o carga.
OnPositionChange	Se produce cuando cambia la posición actual del archivo que está en reproducción.
OnNewStream	Se produce cuando se inicia la carga y reproducción de un nuevo fichero.
OnReadyStateChange	Se produce cuando cambia la propiedad *ReadyState.*

LISTADO 1. Código asociado al control *TMediaPlayer.*

```
procedure TFReproductor.MPWarning(
   Sender: TObject; WarningType,
   Param: Integer;
   const Description: WideString);
begin
 MessageDlg('Error:+IntToStr(Param)
 + #13 + Description, mtError,
 [mbOk], 0);
end;

//Reproduce un fichero
procedure TFReproductor.ReproducirFichero;
begin
  try
    //Reproducir
    MP.Open(Fichero);
  except
    ShowMessage('Error de reproduccion');
  end;
end;
```

un componente *TTimer* cuya propiedad **Interval** vale 30000 (30 segundos). Se puede ver el código asociado a los botones y al *TTimer* en el Listado 2, también incluido en el *CD-ROM.* Conviene indicar que sólo resultará efectivo si nuestro ordenador dispone de una placa base *ATX.* En otro caso, dejará el ordenador mostrando el mensaje **Ahora puede apagar su equipo.**

El control es capaz de reproducir listas de MP3's (archivos M3U)

Este botón es del tipo *Check* (con la propiedad *Style* igual a *tbsCheck*), por lo que al pulsarlo se quedará "hundido", lo que nos indica que el apagado automático se encuentra activado.

INSTALACIÓN

Después de concluir cualquier aplicación conviene recordar que para su instalación en otros sistemas (donde no se dispone de *Delphi* o de alguno de los controles empleados) es necesario distribuir junto al ejecutable aquellos controles y bibliotecas *(DLLs)* utilizados. En nuestro caso, además del ejecutable debemos incluir el fichero **msdxm.ocx.** Además, mediante un instalador de aplicaciones, o similar, debemos registrar el control *ActiveX* en el sistema.

Con una simple línea de código podemos apagar el ordenador

Una vez realizados estos pasos, lanzaremos nuestra aplicación sin más problemas. El propio *Delphi* incluye el programa *InstallShield Express* que permite crear instalaciones que se encargan de todo el proceso



LISTADO 2. Código asociado a los botones y al *TTimer*

```
procedure TFReproductor.ApagarBtnClick(
  Sender: TObject);
begin
 if ApagarBtn.Down then
    begin
     if InputQuery('Introduzca la
         hora', 'Apagar el sistema a
         las....', HoraIntroducida) then
       Timer.Enabled:=True;
      else
        ApagarBtn.Down:=False;
      end
  else
    // Botón no pulsado,
   // desactivar Timer
    Timer.Enabled:=False;
end;

procedure TFReproductor.TimerTimer(
  Sender: TObject);
var HoraActual:string;
begin
 HoraActual:=TimeToStr(Time);
 //Le quitamos los segundos.....
 Delete(HoraActual,6,3);
 if HoraActual=HoraIntroducida then
   ExitWindowsEx(EWX_SHUTDOWN+
     EWX_POWEROFF,0);
end;

procedure TFReproductor.ReproFichBtnClick(
    Sender: TObject);
begin
 if AbrirDlg.Execute then
     ReproducirFichero(AbrirDlg.FileName);
end;
```

de una forma automática y transparente para el usuario.

Durante la etapa de desarrollo, lo mejor es instalar en nuestra máquina (y en aquellas en la que vayamos a realizar las pruebas) la aplicación *Windows Media Player,* puesto que ésta instala y registra el componente *ActiveX* en el sistema. Se puede encontrar en la *web* de *Microsoft:-* (http://www.microsoft.com/windows/mediaplayer/en/default.asp).

EL PRÓXIMO MES

En la próxima entrega, comenzaremos por ampliar el reproductor multimedia. Vamos a añadir, entre otras cosas, la posibilidad de reproducir una lista de ficheros, la capacidad de aceptar los ficheros que se arrastren sobre la aplicación, controles personalizados de volumen, balance, *Play, Stop, Pause,* etc.

En principio nuestro reproductor es sencillo, en próximas entregas lo complicaremos

Además, cambiaremos un poco la interfaz de la aplicación que estamos desarrollando. Por último, comenzaremos a dar unas nociones básicas de MCI, que completaremos en el tercer artículo de la serie.

Sin embargo, a los lectores que deseen investigar por su cuenta, les recomiendo que visiten el sitio *Web* de *Microsoft,* donde podrán encontrar abundante e interesante información sobre el tema de la reproducción multimedia.

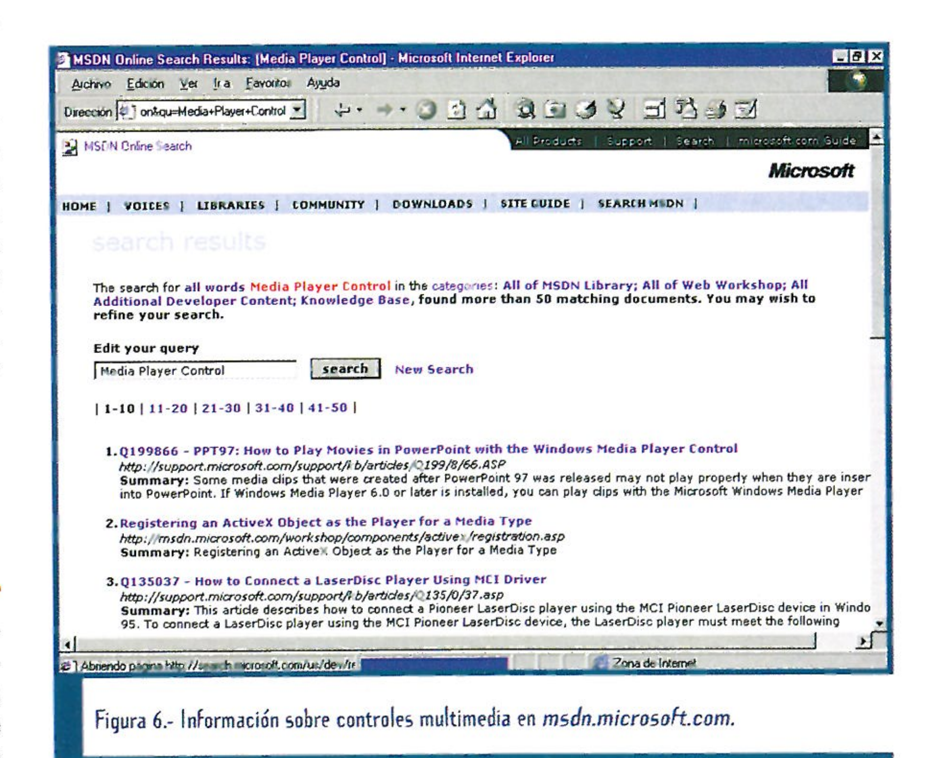


Figura 6.- Información sobre controles multimedia en *msdn.microsoft.com.*

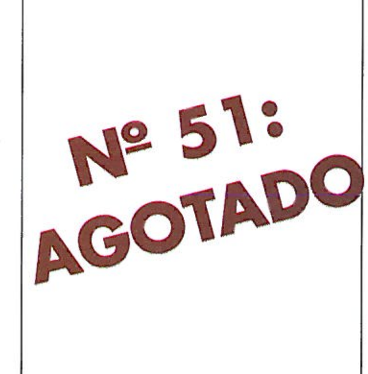
¿TE FALTA ALGÚN NÚMERO DE...?

LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SÓLO PROGRAMADORES













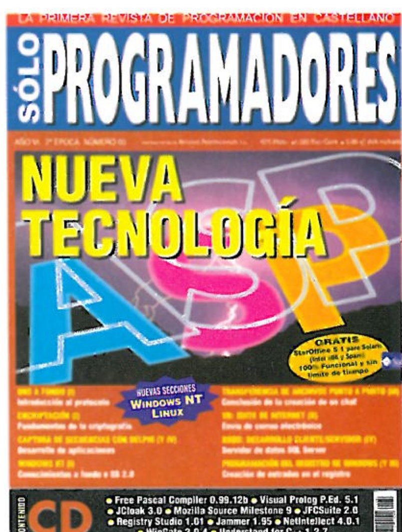












BOLETÍN DE PEDIDOS

Rellene o fotocopie el cupón y envíelo a REVISTAS PROFESIONALES, S.L.
(Revista Sólo Programadores). C/ San Sotero, 5. 1ª Planta. 28037 Madrid
Tlf: 91 304 87 64. Fax: 91 327 13 03

Deseo que me manden los número/s...............................
....
....
NOMBRE Y APELLIDOS:..
....
EDAD:........... PROFESIÓN: ....TFNO: ....
DOMICILIO: ....
CIUDAD: ....
PROVINCIA: ....C.P: ....

❑ **Precio por unidad: 975 pesetas** + 700 ptas. de gastos de envío.

FORMAS DE PAGO:

❑ Giro postal a nombre de REVISTAS PROFESIONALES, S.L
❑ Talón bancario a nombre de REVISTAS PROFESIONALES. S.L
❑ Domiciliación bancaria
❑ Contra reembolso
❑ Transferencia al Banco Popular Español. C/ Valdecanillas, 41. Nº c/c: 0075/1040/43/0600047439

DATOS DE DOMICILIACIÓN:

Banco.. FIRMA
Domicilio...
Nº de Cuenta............|............|............|..............................
Titular..
Fecha...





DNS a fondo (II)
Búsquedas de información

Vicente Antonio Sánchez Werner.
Desarrollador independiente

En el artículo anterior vimos el funcionamiento de *DNS,* su origen y los conceptos teóricos en los que se asentaba. Ahora explicaremos cómo obtener información sobre el *DNS* desde *Linux.*

Como veremos *DNS* no sólo está relacionado con el servicio de nombres, sino que sirve de base para el funcionamiento de otros protocolos, por lo que un mal funcionamiento o una configuración incorrecta provoca una avalancha de fallos.

ACCESO A LA INFORMACIÓN

DNS era una base de datos distribuida y, por lo tanto, hemos de tener en cuenta dos características para entender cómo se organiza la información y cómo buscarla. El primer punto que se debe considerar es que, por ser una base de datos, la información está almacenada en diversos registros y que éstos son de varios tipos.

El segundo punto se refiere a que la información está repartida por un gran número de máquinas, y aunque existe cierto grado de redundancia, el número de ordenadores donde podemos encontrar información es muy reducido.

DNS es una base de datos distribuida, para buscar información debemos acceder a diversos ordenadores

Teniendo en cuenta esto necesitaremos una herramienta para hacer peticiones de los diversos registros y que permita navegar por la jerarquía de ordenadores que conforman el *DNS*.

Aunque hay varias utilidades, tres son las más utilizadas: *nslookup*, *dig* y *domtools*. Las dos primeras están incluidas en la distribución estándar de *BIND* (el servidor *DNS* de referencia). Por su parte *domtools* resulta más cómoda y sencilla de manejar, pero se usa mucho menos ya que no está incorporada en las grandes distribuciones.

NSLOOKUP Y EL DNS

El programa *Nslookup* es una de las aplicaciones de diagnóstico integradas en el paquete *BIND*. Su utilidad reside en la consulta, búsqueda y extracción de información del servicio *DNS*. Por su parte *dig* es similar, pero opera a un menor nivel, más cercano al protocolo, lo que la capacita para todo tipo de funciones, aunque por su naturaleza es más usada en labores de diagnóstico que en otros ámbitos.

Si tenemos en cuenta las dificultades en el uso de *dig* y de *nslookup*,



diremos que es más sencillo iniciar el *DNS* con *nslookup* que con *dig*.

Las utilidades más interesantes para buscar información son nslookup, dig y domtools

En este punto hemos de tener funcionando en nuestra máquina una conexión a *Internet* y haber instalado el paquete *BIND* de nuestra distribución, así como el de utilidades de *DNS*. Estaremos preparados para investigar el *DNS* cuando nos conectemos a *Internet*, para que podamos acceder a otros ordenadores con su nombre y ejecutar *nslookup*.

USANDO NSLOOKUP

El uso de *nslookup* puede realizarse de dos formas diferentes, interactiva y no interactiva. La ejecución interactiva se inicia introduciendo el comando *nslookup* sin parámetros. Así podemos efectuar diversas consultas, y cambiar de servidor *DNS* mientras efectuamos nuestra búsqueda.

La ejecución no interactiva de *nslookup* se lleva a cabo cuando introducimos algún parámetro en la línea de comandos. Sin embargo no permite una búsqueda interactiva y ésta es extremadamente sensible a fallos. Cuando fracasa la búsqueda devuelve un error y se para.

Una buena justificación para que nos centremos principalmente en el uso de *nslookup* en modo interactivo, aunque veamos un ejemplo de cómo se ejecutaría *nslookup* en el otro modo:

```
#nslookup  www.isoc.org
Server:  MASTER.maptel.es
Address:  195.76.10.3
Non-authoritative answer:Name:
info.isoc.orgAddress:
198.6.250.9Aliases:
www.isoc.org
```

Este comando devuelve la dirección *IP* de la máquina **www. isoc. org,** así como los alias de la misma. Una búsqueda más compleja habría requerido el uso de diversos parámetros difíciles de recordar. Así pues, generalmente la utilización del modo no interactivo de *nslookup* queda reservado a su uso desde *scripts*.

La utilización interactiva requiere el aprendizaje de algunos sencillos comandos, pero su manejo es más simple que el uso de parámetros de línea de comandos. Veamos cómo sería la misma operación utilizando el modo interactivo de *nslookup*:

```
#nslookup
Default Server:  MASTER.maptel.es
Address:  195.76.10.3
> www.isoc.org
(La salida es
idéntica al caso
anterior)
```

Como vemos, la ejecución interactiva nos lleva a un *prompt* > desde el que introducimos los comandos. Si tecleamos un nombre de una máquina o de un dominio, automáticamente efectuará la petición de información que esté definida.

En este instante ya hemos realizado nuestra primera consulta al servicio de *DNS*. Una consulta que no sólo devuelve la dirección *IP*, sino que también proporciona los alias con los que se puede acceder a esa máquina, pues puede ser un servidor virtual.

Nslookup es la utilidad más usada pese a no ser ni la más potente ni la más cómoda

Además avisa al usuario de que la respuesta no es autorizada. Esto quiere decir que la información no es del todo fiable ya que se ha recibido de forma indirecta a través de llamadas entre servidores *DNS* efectuadas por el servidor **MASTER.maptel.es.** Si hubiésemos hecho la petición a alguno de los servidores autorizados del dominio **isoc.org** la respuesta sería autorizada.

Este tipo de consulta se denomina de tipo *A*, y devuelve la informa-

```
Archivo   Sesiones   Opciones
[root@localhost /root]# dig

; <<>> DiG 8.2 <<>>
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4
;; flags: qr rd ra; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 13
;; QUERY SECTION:
;;      ., type = NS, class = IN

;; ANSWER SECTION:
.                       3d10h14m39s IN NS  E.ROOT-SERVERS.NET.
.                       3d10h14m39s IN NS  I.ROOT-SERVERS.NET.
.                       3d10h14m39s IN NS  M.ROOT-SERVERS.NET.
.                       3d10h14m39s IN NS  L.ROOT-SERVERS.NET.
.                       3d10h14m39s IN NS  K.ROOT-SERVERS.NET.
.                       3d10h14m39s IN NS  J.ROOT-SERVERS.NET.
.                       3d10h14m39s IN NS  B.ROOT-SERVERS.NET.
.                       3d10h14m39s IN NS  F.ROOT-SERVERS.NET.
.                       3d10h14m39s IN NS  G.ROOT-SERVERS.NET.
.                       3d10h14m39s IN NS  C.ROOT-SERVERS.NET.
.                       3d10h14m39s IN NS  H.ROOT-SERVERS.NET.
.                       3d10h14m39s IN NS  A.ROOT-SERVERS.NET.
.                       3d10h14m39s IN NS  D.ROOT-SERVERS.NET.

;; ADDITIONAL SECTION:
E.ROOT-SERVERS.NET.     4d10h14m39s IN A  192.203.230.10
I.ROOT-SERVERS.NET.     4d10h14m39s IN A  192.36.148.17
M.ROOT-SERVERS.NET.     4d10h14m39s IN A  202.12.27.33
L.ROOT-SERVERS.NET.     4d10h14m39s IN A  198.32.64.12
K.ROOT-SERVERS.NET.     4d10h14m39s IN A  193.0.14.129
J.ROOT-SERVERS.NET.     4d10h14m39s IN A  198.41.0.10
B.ROOT-SERVERS.NET.     4d10h14m39s IN A  128.9.0.107
F.ROOT-SERVERS.NET.     4d10h14m39s IN A  192.5.5.241
G.ROOT-SERVERS.NET.     4d10h14m39s IN A  192.112.36.4
C.ROOT-SERVERS.NET.     4d10h14m39s IN A  192.33.4.12
H.ROOT-SERVERS.NET.     4d10h14m39s IN A  128.63.2.53
A.ROOT-SERVERS.NET.     4d10h14m39s IN A  198.41.0.4
D.ROOT-SERVERS.NET.     4d10h14m39s IN A  128.8.10.90

;; Total query time: 306 msec
;; FROM: localhost.localdomain to SERVER: default -- 195.76.10.3
;; WHEN: Sat Oct  2 01:01:53 1999
;; MSG SIZE  sent: 17  rcvd: 436

[root@localhost /root]#
```

Figura 1.- Ejecución de la utilidad *dig*.

ción contenida en los registros de tipo *A*, *CNAME* y *PTR*, los tres tipos de registros básicos para el funcionamiento de *Internet*.

Un registro de tipo *A* define la asociación entre un nombre y una dirección numérica, siendo el único responsable de la asignación de un nombre a una *IP*. Este registro se pensó para asignar una única *IP* a una única máquina y con una única dirección *IP*. El caso de una máquina con diversas direcciones *IP* o con una interfaz de red que opere con dos o más direcciones *IP* se contempla asignando un registro *A* a cada una de estas direcciones, por lo que el ordenador tendrá más de un nombre por definición. El nombre asignado a una *IP* en este tipo de registros recibe la denominación de canónico o primario.

La información que contiene el DNS se encuentra agrupada en diferentes tipos de registros

El registro *A* definía un caso de relación directa entre una *IP* y un nombre, pero si el nombre solicitado es un alias de la máquina real, la búsqueda en los registros *A* no daría resultado satisfactorio.

Este problema queda resuelto con la búsqueda en los registros *CNAME*, que asignan un nombre primario o canónico a un alias, definiendo la relación entre un alias determinado y una dirección *IP*. La asignación se lleva a cabo de forma indirecta, por facilidad de mantenimiento.

Esto puede verse fácilmente si suponemos que queremos cambiar la *IP* de una máquina hipotética que dispone de diversos alias. Si la asignación de los mismos se hiciese determinando el alias a la dirección *IP*, tendríamos que modificar todos los registros. Sin embargo, si asignamos a cada alias un nombre canónico, no tendremos que modificar ninguno, porque al cambiar la dirección *IP* en el registro *A* no rompemos la relación entre los alias y su nombre canónico. Así nos ahorramos costes de mantenimiento y facilitamos el manejo y administración del servidor.

Finalmente se realiza una búsqueda a través de los registros *PTR*; éstos permiten redirigir la búsqueda dentro de un mismo dominio. Este tipo de registros permite la resolución inversa mediante el dominio **IN-ADDR.ARPA**, posibilitando una búsqueda más rápida.

Si queremos realizar la operación contraria, retomando el artículo anterior, deberíamos construir manualmente un nombre perteneciente al dominio **IN-ADDR.ARPA** en conjunción con la *IP*:

```
> 158.109.8.62
Server:   MASTER.maptel.es
Address:  195.76.10.3
Name:     bbs-ce.uab.es
Address:  158.109.8.62
```

El nombre real de una máquina se conoce también como nombre canónico o primario

Como podemos ver, en lugar de usar un nombre construido por la dirección *IP* de la máquina escrita en orden inverso y con el dominio **.IN-ADDR.ARPA** al final, utilizamos exclusivamente la *IP*. *Nslookup* se encarga de transformarla y de efectuar la búsqueda. Esta operación construye un nombre buscado a través de los registros *A* y *PTR* del servicio *DNS*, ya que los *PTR* van dirigiendo la búsqueda a la máquina que puede realizarla, entre los registros *A* de su dominio.

Como hemos visto en el primer ejemplo, recibíamos una informa-

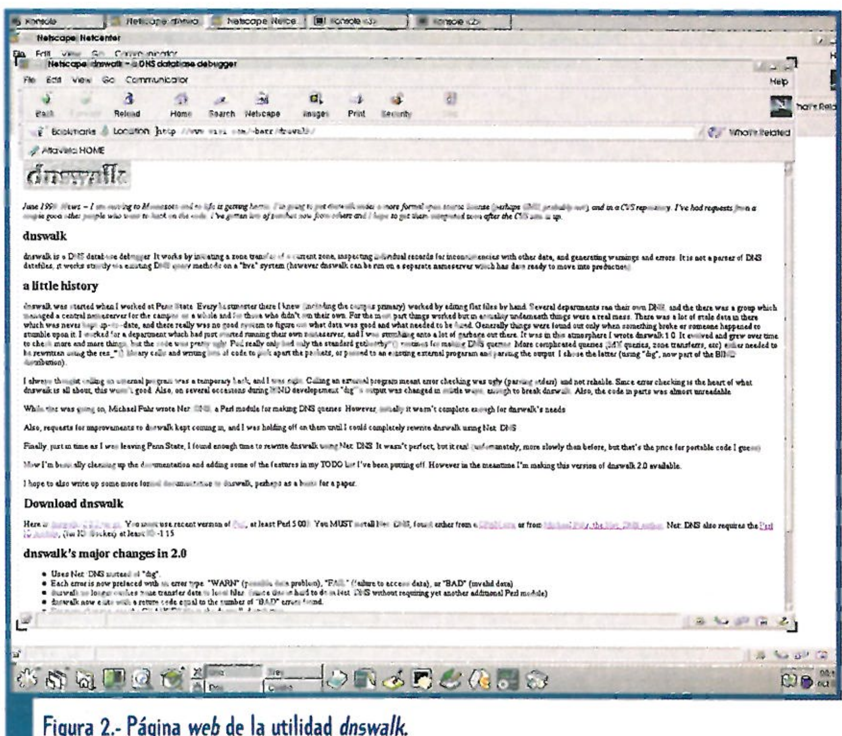


Figura 2.- Página *web* de la utilidad *dnswalk*.



ción calificada como no fiable. Esto se soluciona indicando a *nslookup* que use uno de los servidores de nombres autorizados para el dominio en cuestión. En caso de conocer un servidor autorizado para un dominio, la operación de cambiar el servidor de nombres en *nslookup* se realizaría con la instrucción *server servidor.del.dominio*, como podemos ver a continuación:

```
> server ns.uab.es
Default Server:   ns.uab.es
Address:   158.109.0.1
```

En caso de no conocer ningún servidor de nombres autorizado debemos buscarlo, utilizando *nslookup*. La forma de realizarlo sería la siguiente:

```
[root@localhost /root]# nslookup
Default Server:   MASTER.maptel.es
Address:   195.76.10.3

> set type=NS
>
```

El primer paso es modificar el objetivo de la búsqueda para que devuelva los servidores de nombres autorizados. Esto se realiza con la orden *set type=TIPO_DE_REGISTRO* (*NS* en este caso): hace que se busquen los registros de tipo *NS* contenidos en ese dominio.

Los registros *NS* definen qué máquinas son servidores de nombres autorizados para un dominio determinado. Una vez realizado este paso debemos decirle al programa que nos devuelva los registros contenidos en el dominio que deseamos, escribiendo el nombre del dominio del que queremos obtener un servidor de nombres autorizado.

```
> uab.es
Server:   MASTER.maptel.es
Address:   195.76.10.3
Non-authoritative answer:
uab.es   nameserver =
    wombat.uab.es
uab.es   nameserver =
    sabweb.uab.es
uab.es   nameserver = chico.
    rediris.es
uab.es   nameserver = sun.
    rediris.es
uab.es   nameserver =
    sinera.iiia.csic.es
uab.es   nameserver = NS.uab.es

Authoritative answers can be
    found from:
wombat.uab.es    Internet address
    = 158.109.0.9
sabweb.uab.es    Internet address
    = 158.109.240.7
chico.rediris.es          Internet
    address = 130.206.1.3
sun.rediris.es   Internet address
    = 130.206.1.2
sinera.iiia.csic.es       Internet
    address = 158.109.36.2
NS.uab.es           Internet address
    = 158.109.0.1
```

Figura 2.- Página *Web* donde se mencionan diversas utilidades no comentadas en este artículo.

Una vez que tenemos esta lista podemos elegir uno y configurarlo como servidor de nombres en *nslookup*. En este momento podemos acceder al contenido de los registros de ese dominio.

Existe la posibilidad de listar los servidores DNS para un dominio cualquiera

Por ejemplo podemos listar todos los servidores de nombres existentes y aquellos servidores *DNS* que operan en subdominios dependientes de éste. Esta operación se realizaría con la orden *ls –t NS dominio.deseado*, donde la opción *–t* indica los registros del dominio especificado que deseamos listar. Un ejemplo de esta operación sería:

```
> server ns.uab.es
Default Server:   ns.uab.es
Address:   158.109.0.1

> ls -t NS uab.es
[ns.uab.es]
```

```
$ORIGIN uab.es.
@     2D IN NS sun.rediris.es.
      2D IN NS chico.rediris.es.
      2D IN NS sabweb
      2D IN NS wombat
      2D IN NS ns
      2D IN NS sinera.iiia.csic.es.
iudexeus 2D IN NS csdex001.iudexeus
```

Con estas instrucciones ya podemos navegar por toda la base de datos del *DNS* para obtener la información que deseemos directamente de la fuente. Además podremos conocer datos de la configuración de las propias máquinas o saber qué servicios de *Internet* están configurados en las que forman el dominio.

DNS también incluye información que afecta a otros protocolos

Entre las posibilidades que brinda *nslookup* está el listar todos los registros existentes en un dominio determinado, con lo que podemos tener acceso a todos ellos. Esta operación se realizaría tal y como podemos ver en la Figura 5.

Otra prestación interesante es la posibilidad de conocer los *mail exchangers* de un dominio. El registro *MX*, especifica qué máquinas hay en un dominio capaces de encaminar correo a sus destinatarios, o hacerlo en la dirección correcta. Un *MX* puede ser un servidor *POP3/-IMAP4*, o un mero intermediario en una cadena.

La importancia de poder conocer este tipo de servidores radica en que el uso del *DNS* permite hacer un diagnóstico del funcionamiento del correo dentro de un dominio, pudiendo verificar la correcta configuración de las rutas y la detección de los peligrosos bucles de correo en los que los mensajes ocasionalmente se quedan atrapados. La operación para listar este tipo de registros dentro de un dominio se lleva a cabo así:

```
>server servitec.servitecnet.com
...
> ls -t MX servitec.net
[servitec.servitecnet.com]
$ORIGIN servitec.net.
@  2D IN MX 10 correo.servitec.
     net.com.
>
```

Una posibilidad interesante se basa en los registros de tipo *WKS*. Se usan para describir los servicios que están disponibles en cada uno de los ordenadores registrados en el *DNS*. La naturaleza del campo es opcional, por lo que en muchos dominios no se ha implementado este registro, puede facilitar la tarea a los *hackers*.

Pero en los dominios en que sí está implementado, es sencillo saber qué máquinas poseen servidores *FTP*, *Web*, etc.

Figura 4.- Página *Web* de *domtools*.

Un ejemplo de cómo listar la información de este tipo contenida en un dominio sería:

```
> cica.es
Server:  MASTER.maptel.es
Address:  195.76.10.3
cica.es
    origin = erik.cica.es
         mail addr =
     hostmaster.cica.es
    serial = 1999092900
         refresh = 86400 (1D)
    retry    = 7200 (2H)
         expire   = 2592000 (4w2d)
    minimum ttl = 172800 (2D)
```

Con *nslookup* también podemos acceder a las interioridades del servicio de *DNS*. Existe un tipo de registro llamado *SOA* único en cada zona. Un dominio puede estar dividido en varias zonas, aunque generalmente sólo hay dos por dominio: la correspondiente a la resolución normal de nombres y la que se corresponde con la resolución inversa.

Este registro *SOA* posee la información de la autoridad de ese dominio, entre la que se incluye la dirección de correo de la persona responsable de la zona, el nombre del servidor de nombres primario para la misma, un número de serie y los parámetros de tiempo de refresco, reintentos, tiempo de expiración y otros datos. El acceso a este registro es sencillo:

```
> ls -t SOA maptel.es
[MASTER.maptel.es]
$ORIGIN maptel.es.
@  2D IN SOA    MASTER root (
         1999061701  ; serial
         1D          ; refresh
         2H          ; retry
         4w2d        ; expiry
         2D )        ; minimum
   2D IN SOA    MASTER root (
         1999061701  ; serial
         1D          ; refresh
         2H          ; retry
```




```
4w2d          ; expiry
2D )          ; minimum
```

En principio parece que podemos acceder a cualquier dato que esté introducido en el servicio de nombres, pero en la práctica no es así. Algunos dominios sólo contestan a peticiones de información provenientes de un servidor de nombres, o sólo responden a algunos tipos de búsqueda. Esto es comprensible si tenemos en cuenta que el acceso total al *DNS* puede comprometer seriamente la seguridad de una red.

El uso de registros WKS puede comprometer la seguridad, al facilitar información a intrusos

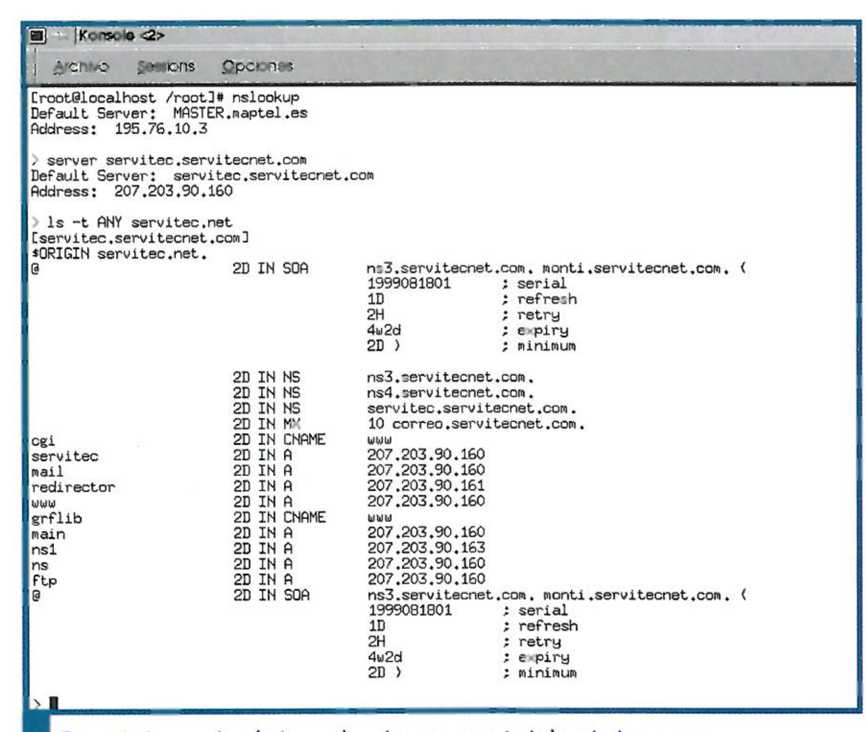


Figura 5.- Imagen de aslookup realizando una operación de listado de registros

Existen otras posibilidades de búsqueda de información en el *DNS*, mediante la petición de otros tipos de registros que podemos efectuar con esta pequeña utilidad, pero se salen del objetivo de este artículo que es asentar las bases prácticas sobre las que descansa el *DNS* y la búsqueda de información en él.

DOMTOOLS

Las utilidades *domtools* son un conjunto de *scripts* y de pequeños filtros escritos en *awk* que transforman los parámetros que se les indican en llamadas a las utilidades *dig* y *nslookup* cuya salida procesan y a las que dan un formato más legible. Estas utilidades pueden encontrarse en **http://www.domtools.com/DNS/domtools.shtml.**

Las utilidades incluidas en *domtools* están agrupadas en tres niveles diferentes, según su potencia. El primero incluye pequeñas herramientas que efectúan tareas de conversión de formatos, cálculo de máscaras de red y otras pequeñas operaciones que no requieren el uso de un servidor de nombres. El segundo nivel lo integran herramientas que realizan búsquedas simples de registros en el *DNS*, de forma mucho más sencilla que *nslookup*.

Domtools son utilidades que ofrecen mayor potencia y comodidad que nslookup

El último nivel está compuesto por herramientas que efectúan operaciones más complejas, que requieren un espacio de tiempo más largo que las anteriores para operar, incluso nos proporciona una lista de los servidores *DNS* que deberían ser autorizados según el dominio inmediatamente superior, o la que genera automáticamente archivos *host* para un dominio.

El uso de estas herramientas es muy simple, ya que la mayoría sólo necesita un nombre de máquina o un dominio para funcionar y, opcionalmente, la dirección de un servidor de nombres. Esto hace que su uso sea más cómodo, pero que a la vez no podamos tener la misma flexibilidad que teníamos con *nslookup*.

La mayoría de las herramientas de primer nivel no aportan nada al conocimiento del *DNS*, ya que no interactúan con los servidores. Son una ayuda para procesar la información obtenida por las utilidades del segundo y tercer grupo.

Las utilidades del segundo grupo son, en mi opinión, las más interesantes de cara a la recogida de información del servicio *DNS*. La primera de ellas es *ns,* su sintaxis:

```
#ns <dominio> [opcionalmente un
     servidor de nombres]
```

Con esta utilidad conseguimos un listado de todos los servidores

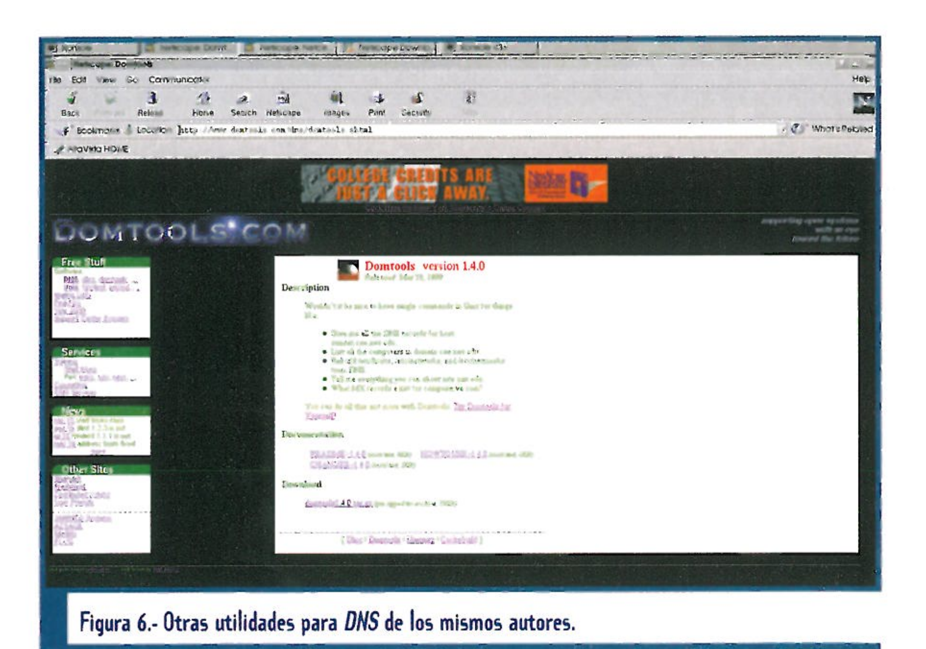


Figura 6.- Otras utilidades para *DNS* de los mismos autores.

de nombres autorizados para el dominio especificado y, como podemos ver, su manejo es mucho más sencillo que operar *nslookup*.

La siguiente utilidad interesante es *root*, que lista todos los servidores de nombres pertenecientes al dominio raíz, por lo que podemos saber exactamente cuántos existen sus nombres. Otras utilidades interesantes son *SOA* (que lista el registro *SOA* para un dominio dado) o *any* (que lista todos los registros referentes al dominio o máquina especificada). La sintaxis de ambas es:

```
#SOA <dominio>
#any <dominio>
```

En el tercer nivel de utilidades cabe destacar *hosttbl*, *siteinfo* y *soalist*. La primera de ellas genera un fichero *hosts* para un dominio dado, recorre todo el dominio generando líneas con información en un formato compatible con el archivo *hosts*. Esta pequeña utilidad es interesante porque permite crear un caché para que las máquinas de nuestra red no dejen acceso a las demás.

Pensemos que si utilizasen exclusivamente el *DNS* para la resolución de nombres, y éste dejase de funcionar, no se podrían crear nuevas conexiones. La sintaxis de esta utilidad requiere varios parámetros; el primero es opcional e indica el número de alias que debe recoger, además del nombre canónico de cada máquina. El segundo es el dominio del que queremos construir la tabla.

```
#hosttbl <nº de alias> <dominio>
```

Este comando daría una salida detallada con información del dominio, pero no crearía el fichero. Para hacerlo deberemos redirigir la salida del programa a un archivo que recogerá los datos generados por la utilidad. A continuación se muestra un ejemplo práctico:

```
#hosttbl 2 jazzfree.com > jazz.txt
```

La segunda utilidad, *siteinfo*, realiza una serie de consultas sobre el dominio especificado y muestra los resultados. El equivalente a este comando en *nslookup* requeriría unos **10** tipos de consultas diferentes, además de tener que usar la utilidad *dig* para efectuar consultas adicionales, que *nslookup* no es capaz de realizar. La sintaxis de este comando sería:

```
#siteinfo dominio.deseado
```

Finalmente, *soalist* se encarga de buscar con la ayuda de los servidores de nombres del dominio inmediatamente superior al solicitado, cuáles deberían ser los servidores de nombres autorizados para éste y lista los registros *SOA* de los mismos, de modo que podamos contrastar los datos de los diferentes servidores y ver si coinciden. La sintaxis de la utilidad sería:

```
#soalist dominio.deseado
```

Con domtools podemos crear de forma sencilla un fichero hosts con toda la información de un dominio dado

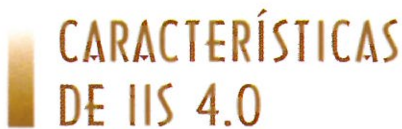
Como podemos comprobar, *domtools* es un paquete de utilidades muy interesante de cara a buscar información y a diagnosticar un servidor de *DNS*. Téngase en cuenta que las utilidades que he mencionado son sólo unas pocas de entre las que componen este extenso paquete, pero abarcan los ámbitos más interesantes.

RESUMEN

En este artículo hemos aprendido a buscar información en el *DNS* y algunas de las utilidades que podemos usar para esta tarea; en el camino hemos visto cómo se organiza la información en el *DNS* y qué registros hay, para qué sirven y qué datos pueden contener. En el próximo artículo veremos la forma de implementar el *DNS* en nuestra máquina *Linux*.



Windows NT 4.0 e IIS (y II)



Jordi Agost Moré
Profesor de Programación/Multimedia de la Universidad de Lleida.

En el anterior artículo vimos las principales diferencias entre *Windows* NT *Server* y *Windows NT WorkStation*, así como las principales características de *Internet Information Server 4.0*, incluido dentro de *Windows NT Option Pack 4*. A continuación repasaremos sus principales características.

CARACTERÍSTICAS DE IIS 4.0

Vamos a ver un resumen de las principales características que incluye el producto *Internet Information Server 4.0*, entre las que destacan:

- Protección contra errores.
- Compatibilidad con *Java*.
- Múltiples sitios *Web* en una sola *IP*
- Estándares de *Internet*.
- Administración y control del contenido.
- Páginas Activas de Servidor (*ASP*)
- Cola de mensajes integrada.
- Administración automatizada.

AJUSTES GENERALES

Si somos los creadores de un sitio *Web* de gran éxito, puede ocurrir que todo nuestro esfuerzo no sirva para nada si en el momento en que empiece a crecer el número de consultas todo el sistema se viene abajo por una sobresaturación de visitas. Vamos a ver a continuación las bases que harán que nuestro servidor funcione a la perfección.

No obstante, antes de intentar esta optimización es importante conocer los puntos claves sobre los cuales se fundamenta. Conoceremos cómo se encuentra el estado de nuestro servidor si lo examinamos bajo tres conceptos muy importantes:

- Carga del procesador: si nuestro procesador se encuentra muy ocupado el rendimiento global puede disminuir. Suponiendo que no podamos disminuir dicha carga siempre podremos incorporar un procesador más rápido o múltiples procesadores.

- Uso de la memoria: su uso tiene una incidencia directa sobre el rendimiento global.

- Uso de recursos: consideraremos recursos los grupos de elementos como por ejemplo el espacio de almacenamiento, la velocidad de conexión, etc.



CONSIDERANDO EL RENDIMIENTO DE IIS

Si bien es importante tener bien configurado el servidor para así optimizar el funcionamiento global, no es menos importante el hecho de configurar *Internet Information Server*. Algunos aspectos que se deben tener en cuenta podrían ser:

- Limitación del ancho de banda: Algunos servicios de red y *Webs* consumen más conexiones que otros. Controlando el ancho de banda de algunos sitios podemos desviar más ancho de banda a otros.

- Limitación de las conexiones: si nuestro servidor tiene muchas visitas deberemos plantearnos el hecho de limitar su número posible, asegurando así un mejor servicio a los usuarios conectados.

Figura 1.- Monitor de rendimiento de *Windows NT*.

Figura 2.- Visualizador de sucesos de *Windows NT*.

- Limitación de los servicios: deberemos tener en cuenta que diferentes servicios consumen diferentes recursos. Por ejemplo los servicios *FTP* consumen muchos menos servicios que el servicio *Web*. Teniendo en mente dichos aspectos podemos planificar de manera correcta nuestro servidor y evitar así una sobrecarga innecesaria.

AJUSTANDO WINDOWS NT SERVER

El primer paso para ajustar nuestro servidor *Windows NT* deberá ser mediante el uso de algunas herramientas que lleva incorporadas.

MONITOR DE RENDIMIENTO

Es una de las primeras herramientas que debemos examinar para el buen funcionamiento de nuestro servidor *Web*. Esta herramienta extremadamente configurable permite, a través de gráficos, conocer cada aspecto de nuestro servidor. Asimismo también podemos evaluar el rendimiento de otras máquinas conectadas a la red.

Utilizando el monitor de rendimiento podemos, de una forma fácil, añadir los recursos o estados del sistema que deseemos vigilar. También cuenta con un sistema de alertas que puede llegar a ejecutar un determinado programa, o lanzar un aviso por la red, en el caso en que una alerta se disparara (por ejemplo nos estamos quedando sin memoria).

También ofrece la posibilidad de grabar toda la actividad de nuestro servidor para analizarla a posteriori.

VISUALIZADOR DE SUCESOS

Se instala automáticamente cuando instalamos *Windows NT* y *WorkStation*. Se utiliza para grabar sucesos y auditar situaciones en nuestra máquina que pueden ser objeto de problemas y que por lo tanto requieren nuestra atención. Los sucesos que no requieren una atención inmediata por parte del usuario son simplemente grabados, mientras que los críticos son también grabados pero se le muestran al usuario a través de un mensaje en pantalla.

Normalmente los eventos se encuentran clasificados en tres apartados:

- Sistema: eventos relacionados con los *drivers* y componentes del sistema.

- Seguridad: se graban los eventos que pueden indicar una brecha en el sistema de seguridad de nuestro sistema. Podemos

grabar los eventos exitosos o los fallidos relacionados con la seguridad del sistema.

- Aplicaciones: se graban los errores con los que se han encontrado las aplicaciones.

Además para cada línea se grabará la fecha y hora en que se ha producido el evento, la fuente que lo ha producido, la categoría a la que pertenece, el número de evento, si un usuario está relacionado con él, su nombre, el ordenador donde ha ocurrido, y con qué color se marcará la clasificación del evento (dentro de los grupos de error, aviso, información, auditoría correcta o fallo en auditoría).

Ya que el Visor de sucesos puede llegar a crear un archivo de *log* muy grande, deberemos tener en cuenta que éste permite limitar el tamaño del fichero o en su defecto los sucesos que queremos escribir.

OTRAS APLICACIONES

PLANIFICADOR DE TAREAS

Otra herramienta muy útil dentro de *NT 4* es el Planificador de tareas o *Task Manager*, que muestra todas las tareas y los hilos de ejecución presentes en nuestra máquina, así como los recursos que utilizan. Además permite cambiar entre sí las tareas, lanzar otras nuevas o finalizar procesos.

VISUALIZADOR DE APLICACIONES

Dicho visualizador permite ver la lista de todas las aplicaciones que se están ejecutando en nuestro ordenador y su estado actual.

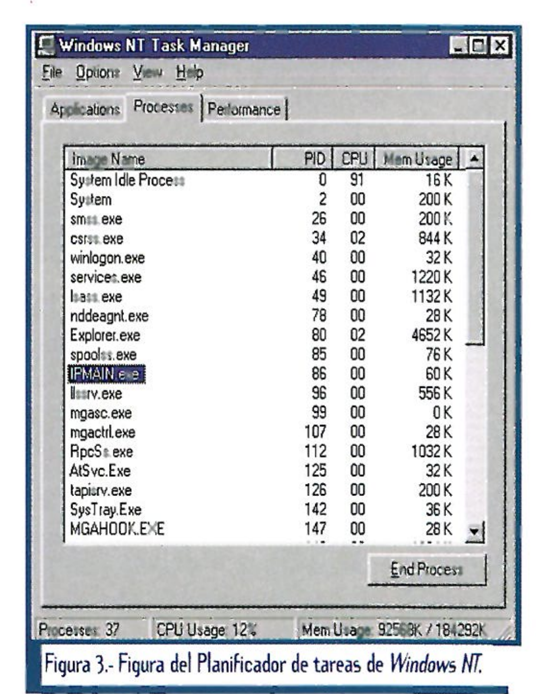


Figura 3.- Figura del Planificador de tareas de *Windows NT*.

VISUALIZADOR DE PROCESOS

Cada programa que se ejecuta en nuestro ordenador está dividido en procesos, algunos de ellos visibles y la mayoría invisibles para el usuario. Dicho visualizador permite finalizar un determinado proceso, controlar su prioridad en tiempo real (respuesta inmediata), alta (mayor preferencia que el límite establecido como normal), normal (se ejecuta sin ningún gasto adicional de procesos), baja (para tareas que no son relevantes y se ejecutan en *background*).

VISUALIZADOR DE RENDIMIENTO

Este visualizador no posee interacción, simplemente muestra el estado de nuestra máquina, incluyendo el porcentaje de uso de la *CPU*, el uso de la memoria, el número total de procesos y de hilos.

Hasta aquí todas las herramientas que hemos visto sirven para monitorizar nuestro servidor y para tomar decisiones de acuerdo con el estado que presenta. Dichas decisiones se dirigirán al manejo de recursos, y a veces serán el indicativo necesario para decidir incrementarlos. Las áreas que monitorizar y de las que debemos hacer un seguimiento comprenden los siguientes campos:

- Almacenamiento: si necesitamos aumentar nuestra capacidad podemos añadir más discos a nuestro servidor o mirar otras soluciones como *arrays* de discos *RAIDS*.

- Memoria: si nuestro servidor incrementa su carga, podemos incrementar su rendimiento aumentando la cantidad de memoria *RAM*, dando así un servicio más rápido y a más gente.

- Procesador: Si el aumento de memoria no ha solucionado nuestro problema podemos decantarnos por adquirir un procesador más rápido o por una solución multiprocesador.

- Distribución de carga: *NT* permite distribuir nuestros servicios con otros servidores de *Windows NT* que se encuentren dentro de su mismo dominio de la red (no confundir con los dominios de *Internet*), para así por ejemplo atender a diferentes bases de usuarios.

CLUSTERING

Suponiendo que las soluciones anteriores no nos hayan dado ningún resultado la solución puede encontrarse en el *clustering* que consiste en la capacidad de agrupar varios servidores para compartir la carga de trabajo.



INCREMENTANDO EL RENDIMIENTO DE IIS

Aunque las características que hemos comentado son muy importantes también podemos incrementar el rendimiento global de nuestros servicios *Web* a través *Internet Information Server* 4.0, que ha hecho grandes progresos para mejorar el aspecto del rendimiento, tanto en lo que se refiere a modificaciones globales de los sitios Web como a sitios individuales.

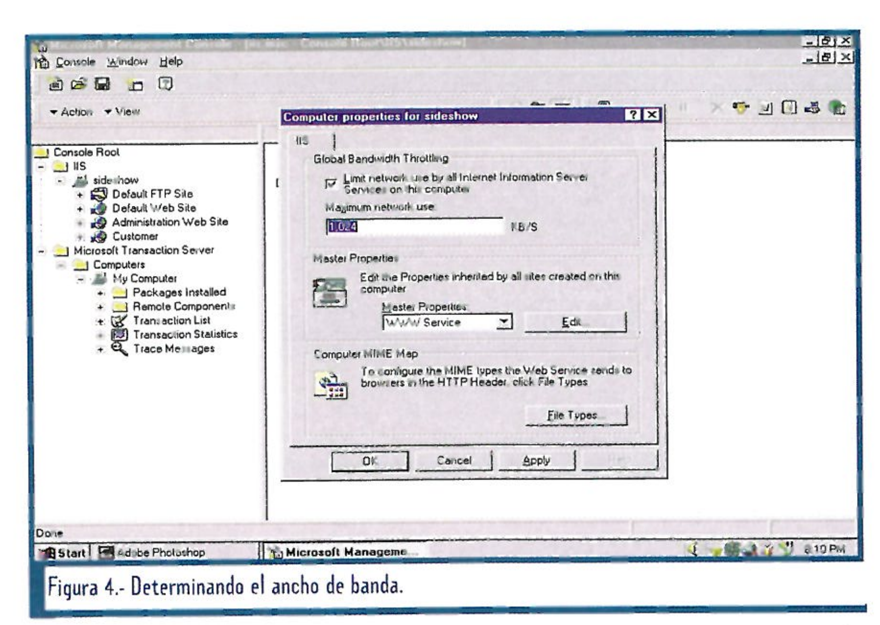


Figura 4.- Determinando el ancho de banda.

RENDIMIENTO GLOBAL DEL SERVIDOR

IIS 4.0 permite definir parámetros de funcionamiento de todo el servidor. Dichos parámetros tienen efecto para cada sitio *Web* y *FTP* que tengamos alojados en nuestro servidor. Para modificar los parámetros globales de *Internet Information Server* deberemos seguir los pasos que detallamos a continuación.

ANCHO DE BANDA

En primer lugar deberemos pulsar con el botón derecho sobre el servidor en el árbol del *MMC* y seleccionar las opciones de *Propiedades*.

Para limitar el máximo de ancho consumido por todos los servicios de *IIS* deberemos seleccionar la caja de chequeo de límite de uso de la red. Entonces se nos activará la caja de texto donde podremos poner nuestro límite siempre expresado en *Kb/s*.

SERVICIOS MAESTROS

También seleccionando el servicio adecuado desde la caja desplegable de propiedades maestras y editándolas podemos controlar los servicios maestros para los servicios de *Web* o de transmisión de ficheros o *FTP*. Con los servicios maestros podremos definir parámetros para todos los sitios *Web* que tengamos en nuestro servidor. Hay cuatro limitaciones principales que modificar, que pueden también ser modificadas individualmente para cada *Web*. Dichas características son:

- Limitación de las conexiones: estableciendo aquí un límite podemos evitar que nuestro servidor caiga si en un determinado momento hay un enorme aumento del número de visitantes. Asimismo también podemos definir el *time-out* de nuestro *Web*. Dicho tiempo servirá para saber cuándo hemos de clausurar una sesión de un visitante después de un error de red.

- Estados de rendimiento: podemos afinar nuestro sitio *Web* para una determinada audiencia. Así podemos incrementar la prioridad de un sitio *Web* determinado y decrementar la de otro que no sea tan visitado.

CONFIGURACIÓN DE LA CONEXIÓN

Dentro del protocolo *HTTP* se considera que una conexión está siempre abierta. Esto puede afectar drásticamente al rendimiento general del servidor, ya que en cierta manera estamos sobrecargándolo con conexiones que puede ser que ya no sea necesario tener activas.

PROPIEDADES PARA EL SERVIDOR DE FTP

Las propiedades globales para el servidor de ficheros o *FTP* son bastante más sencillas. Tan sólo se limitan al número máximo de conexiones que nuestro servidor permitirá. Si limitamos ese número, lo que estamos haciendo en realidad es limitar el número individual de

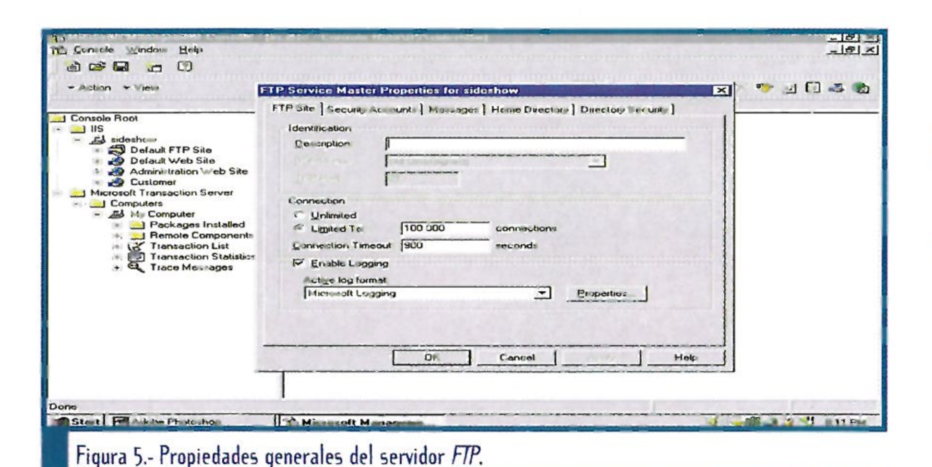


Figura 5.- Propiedades generales del servidor *FTP*.

sesiones que se pueden abrir. Para establecer este valor deberemos seguir los siguientes pasos:

- Abrir la caja de diálogo de las propiedades generales del sitio *FTP*.
- El grupo de conexión funciona como en el servicio de *Web*. Para limitar el número total de usuarios seleccionaremos el *radio button*.
- Aparecerá una caja de texto en la que introduciremos el número máximo de conexiones permitidas.
- Modificaremos, si lo deseamos, el tiempo time-out del servicio en la caja de texto *Connection Time Out*.
- Aceptaremos los cambios.

OTRAS CONSIDERACIONES

CONSIDERANDO LOS FICHEROS

El tema de tratamiento de los ficheros es de gran importancia ya que deberemos tener en cuenta que todo lo que se envía o se recibe de un sitio *Web* es un fichero. El tiempo que tarda en servirse un fichero y su tamaño están relacionados aunque no necesariamente de una forma directa. Por ejemplo si un fichero es pequeño requiere un *software* adicional para procesarse.

MODIFICANDO SITIOS WEB INDIVIDUALMENTE

Cuando modificamos las propiedades maestras para un determinado servicio, todos los sitios son modificados a la vez. Además en la consola gráfica también podemos hacer los cambios individuales para cada uno de estos sitios si así lo deseamos, aunque deberemos tener en cuenta que los cambios de propiedades comunes son excluyentes entre sí.

CONSIDERACIONES SOBRE EL SITIO WEB

No debemos olvidar en ningún momento que la mayoría de usuarios poseen una conexión lenta. Tampoco debemos olvidar los estudios que nos dicen que la mayoría de usuarios no espera más de 30 segundos para la visualización de una página. Así que si sobrecargamos nuestras páginas con gráficos de gran tamaño o con enormes ficheros multimedia lo único que lograremos es sobrepasar este límite y correr el riego de que nuestro usuario no espere a visualizar la página.

PLANIFICANDO LA CONEXIÓN

A través de una cuidadosa planificación podemos hacer que nuestro sitio *Web* minimice los tiempos muertos e incremente su eficacia. Ambas cosas harán que el usuario final se encuentre más a gusto con nuestro sitio *Web*. Veamos unas directrices a tener en cuenta:

- Utilizar los mismos gráficos: siempre y cuando sea posible tenemos que reutilizar nuestros gráficos ya que así aprovecharemos la memoria caché del navegador del usuario y la velocidad global se incrementará.
- Tener en cuenta las conexiones lentas: si podemos diseñar un sitio *Web* de forma que el usuario al conectarse pueda elegir entre una conexión rápida y una conexión lenta, sin lugar a dudas contentaremos a los usuarios que no posean tantos recursos.
- *Add-ons*: si rellenamos nuestro sitio *Web* con contenidos *Java* o *ActiveX*... nuestras páginas *Web* aumentarán de calidad y de vistosidad, pero debemos tener en cuenta que también tardarán más en visualizarse.
- Sencillez: deberemos enviar al usuario tan sólo lo que necesite.
- Actualización: a medida que nuestro sitio *Web* crece no debemos olvidar actualizar las páginas más antiguas, tanto en contenido como por ejemplo en tipos de ficheros obsoletos.



Tabla 1. Códigos de error más comunes en el protocolo HTTP.

Status	Message
400	Bad Request
401.1	Unauthorized:Logon Failed
401.2	Unauthorized: Logon Failed due to server configuration
401.3	Unauthorized: Unauthorized due to ACL on resource
401.4	Unauthorized: Authorization failed by filter
401.4	Unauthorized: Authorization failed by filter
401.5	Unauthorized: Authorization failed by ISAPI/CGI app
403.1	Forbidden: Execute Access Forbidden
403.1	Forbidden: Execute Access Forbidden
403.2	Forbidden: Read Access Forbidden
403.3	Forbidden: Write Access Forbidden
403.4	Forbidden: SSL required
403.5	Forbidden: SSL 128 required
403.6	Forbidden: IP address rejected
403.7	Forbidden: Client certificate required
403.8	Forbidden: Site access denied
403.9	Access Forbidden: Too many users are connected
403.10	Access Forbidden: Invalid Configuration
403.11	Access Forbidden: Password Change
403.12	Access Forbidden: Mapper Denied Access
404	File Not Found405Method Not Allowed
406	Not Acceptable
407	Proxy Authentication Required
412	Precondition Failed
414	Request-URI Too Long
500	Internal Server Error
501	Not Implemented
502	Bad Gateway

PÁGINAS PERSONALIZADAS DE ERRORES

Vamos a ver a continuación, una vez conocidos los diferentes ajustes que podemos realizar en nuestro servidor y en nuestro programa, una de las configuraciones más útiles que nos servirán para dar mensajes personalizados de error.

En *IIS 4.0* podemos devolver páginas personalizadas para cada uno de los errores de cada uno de los sitios *Web*. Así en vez del típico mensaje:

"404 File Not Found"

El usuario podría tener una lista opcional de *links* con la cabecera de nuestra compañía y un mensaje de error mucho más profesional y personalizado.

Para cada error estándar del protocolo *HTTP* (errores 403, 404, 504) se devuelve un mensaje al navegador. Es en este punto donde *IIS 4.0* permite devolver una página *Web* específica con más información del error, o bien un fichero predeterminado o en su defecto el error.

Cuando se instala *IIS 4.0* se instalan los ficheros de errores en la dirección: *C:\winnt\help\common*.

En la Tabla 1 aparecen los errores más comunes que se producen en los sitios *Web* junto con su código, y a los que les podremos dar una respuesta diferente.

El paso más simple para lograr nuestro objetivo consiste en modificar directamente los ficheros guardados en la ubicación comentada anteriormente (ya que son ficheros *HTML* normales) para, por ejemplo, añadirles una cabecera de nuestra compañía.

MÚLTIPLES SITIOS WEB

Ya que una de las características que permite *IIS 4.0* es la posibilidad de tener múltiples sitios *Web* en una misma máquina. Veamos a continuación cómo personalizar los errores.

- Abrimos *MMC*.
- Escogemos el sitio *Web* para el que queremos personalizar los mensajes de error.
- Seleccionamos *Propiedades* desde el menú desplegable.
- Seguidamente escogemos la pestaña de errores personalizables (*Custom Error*).
- De la lista de errores *HTTP* escogemos los que deseamos "mapear".
- Editamos las propiedades.
- Cambiamos el tipo de mensaje hacia una *URL*.
- Aceptamos los cambios.

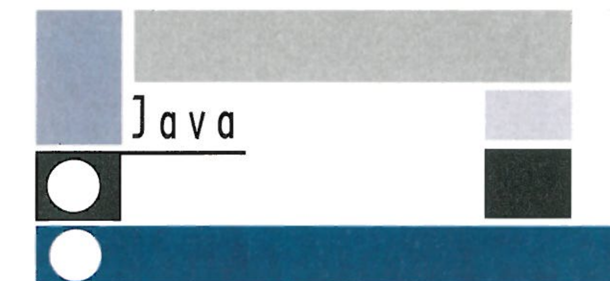
Dicha *URL* debe estar dentro del ámbito del sitio *Web* y además debe estar referenciada por la raíz virtual, como por ejemplo:

\error\404err.htm

Una idea que puede aplicarse sobre todo lo anterior es también "mapear" la respuesta de los errores más comunes a la dirección principal del *Web*.

MENSAJES DE ERROR DINÁMICOS

Otra posibilidad consiste en devolver una página activa de servidor (*ASP – Active Server Pages*) en vez de una página estática, ya que todo lo que tenemos que hacer es crear una página *ASP* y "mapearla" según el procedimiento explicado anteriormente. Un punto a tener en cuenta es que esta técnica tan sólo deberíamos emplearla en los mensajes cuyo rango de error se encuentra entre 400-499 ya que los que se encuentran entre 500-599 indican un error del servidor y puede ser que la página *ASP* no se devuelva correctamente.



Criptografía (II)

Javier Sanz Alamillo. Ingeniero de Software.

La utilización de métodos criptográficos modernos permite el desarrollo de aplicaciones que garantizan la seguridad y tratamiento de la información que gestionan. Gracias a la variedad de métodos actuales, adaptar las necesidades a las aplicaciones es una tarea simple y práctica.

Si en el anterior artículo se presentaron los fundamentos de la criptografía y los métodos clásicos más elementales, en éste se mostrarán los fundamentos de la generación de claves y se detallarán los algoritmos más importantes que forman los criptosistemas de clave privada.

GENERACIÓN DE CLAVES

En los criptosistemas actuales, un tema de gran relevancia lo constituye la gestión de las claves que se están utilizando, desde el punto de vista de la generación, almacenamiento, distribución y mantenimiento, de nuestro sistema

Poder contar con buenos sistemas para la creación de claves es muy importante. Así, estas claves deben ser impredecibles y, por tanto, la mejor manera de crearlas es de forma aleatoria (pero equiprobable), mediante el uso de los algoritmos de generación de números aleatorios que se han mostrado.

Veamos a continuación un conjunto de sistemas y métodos utilizados para generar números aleatorios y por tanto posibles claves.

GENERADORES DE RUIDO ELÉCTRICO

El ruido eléctrico, aunque es un fenómeno indeseable en los amplificadores electrónicos, puede ser utilizado como fuente aleatoria de señales. Así, por ejemplo el ruido producido al calentar una resistencia es un buen generador de secuencias aleatorias de *bits*.

Cuando se calienta una resistencia, ésta produce una ruido que se amplia con un amplificador de banda ancha y con ello, se alimenta a una puerta lógica de la que salen impulsos, que tratados mediante un contador ofrecen una secuencia que puede ser utilizada como una clave binaria. Aunque de reducido coste no suele ser utilizado debido a su complejidad práctica.

GENERADORES CON DESPLAZAMIENTO RDRL

Se denomina así a un conjunto de generadores de *bits* basados en *Registros de Desplazamiento de Realimentación Lineal (RDRL)*. Normalmente son el tipo de generadores de desplazamiento más comúnmente utilizados.

Se trata de un registro, que denominaremos *R*, que consta de *n* posiciones binarias y de una secuencia *S* igualmente binaria,

```
R = { r1 , r2 , r3 ... , rn-1 ,
    rn }
S = { sn , sn-1 , sn-2 ... , s2 ,
    s1 }
```

que se suma mediante una operación de *XOR* (o-exclusivo) con la *R* *bit* a *bit*, y cuyo resultado se alma-



cena en r_n en el instante siguiente, previo desplazamiento de los *bits* { r_1 , r_2 ,..., r_{n-1} , r_n}, una posición a la derecha. Este proceso se puede representar así:

```
ri(t+1) = ri(t)      i = 1 ... (n-
     1)
rn(t+1)= S.R = si(t).ri(t)≈..≈
     sn(t).rn(t)
```

donde el símbolo ≈ equivale a la operación *XOR* citada.

Los *RDRL* generan cadenas de *bits* con un periodo de 2^n*-1*, a partir de la secuencia *S*, que hace que *R* pase por las 2^n*-1* secuencias.

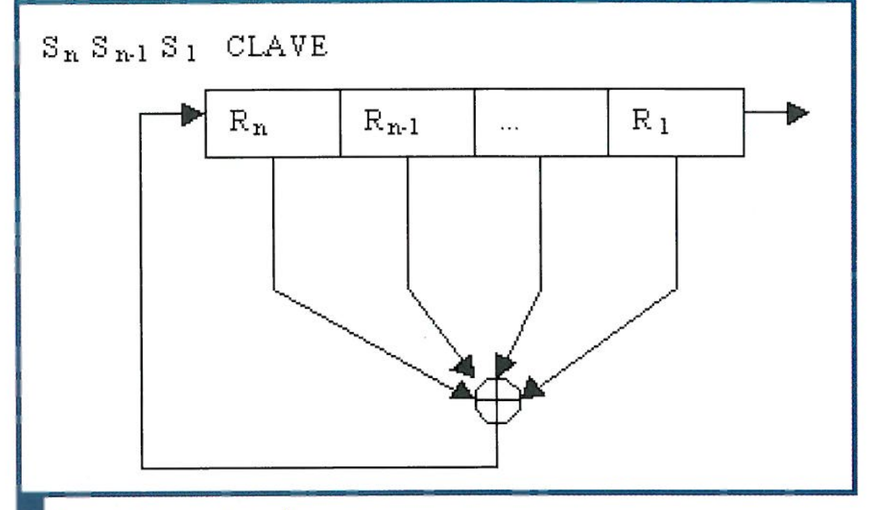


Figura 1.- Registros de desplazamiento.

Los generadores mediante desplazamiento son una opción sencilla para generar secuencias aleatorias de bits

Veamos un ejemplo, en el que tomaremos S = { 1, 0, 0, 1} y R = {0, 0, 0, 1}. Aplicando la suma "o exclusiva" y los desplazamientos como se ha descrito, se obtendría la siguiente secuencia:

```
t = 1. 0001
    2. 1000
    3. 1100
    4. 1110
    5. 1111
    6. 0111
    7. 1011
    8. 0101
    9. 1010
   10. 1101
   11. 0110
   12. 0011
   13. 1001
   14. 0100
   15. 0010
```

Se toma como clave el *bit* de menor peso (el situado más a la derecha) de cada secuencia generada, luego la clave sería:

```
100011110101100
```

Conviene que el lector intente implementar este método de generación en *Java*, intentando obtener un medio por el que no se repita la secuencia generada en el periodo anteriormente descrito.

GENERADORES DE NÚMEROS ALEATORIOS POR ORDENADOR

Se basa en la utilización de fórmulas que producen una secuencia de números con un período bastante grande, pero sin ser definitivamente aleatorios puros, lo que quiere decir que en cierta medida son predecibles. Los generadores de números más utilizados son:

- Congruencial multiplicativo mixto.
- Congruencial multiplicativo.
- Serie de *Fibonacci* o congruencial aditivo.

1.- Congruencial multiplicativo mixto

Con este sistema se obtiene una sucesión de números mediante la aplicación de la siguiente fórmula:

```
Zi = a.Zi-1 + c ( mod m ),      (
     i = 1 .. n )
```

Se tiene que Z_0 es el valor inicial o semilla, *a* y *c* son dos constantes y *m* un número muy grande, del mayor tamaño que puede representar el ordenador y que determina la longitud máxima de cada número en la sucesión.

Hoy es fundamental disponer de medios para crear secuencias aleatorias de números

Debido a que cada número se determina por su predecesor y todos están condicionados al valor inicial o semilla, la secuencia generada no es totalmente aleatoria. A pesar de ello, la distribución es uniforme y los números son independientes los unos de los otros.

Imagine un tipo de ataque basado en la creación de los valores

iniciales más probables y a partir de ahí generar toda las secuencias posibles hasta dar con una elegida.

2.- Congruencial multiplicativo

Este método se deriva del anterior y se toma $c = 0$, por tanto:

```
Zi = a.Zi-1 ( mod m ),     ( i =
   1 .. n )
```

En ambos casos, hay que definir otra constante importante a la hora de considerar el valor para las constantes a y c.

Se define P como la longitud de la palabra del ordenador expresada en *bits*. Con ella se puede definir a su vez la constante a de la siguiente forma:

```
a = 2(p/2)+1 + 3
```

De ahí se deduce que m toma como valor 2^p .La constante a es importante porque es la determinante de los períodos de repetición y genera coeficientes de correlación pequeños.

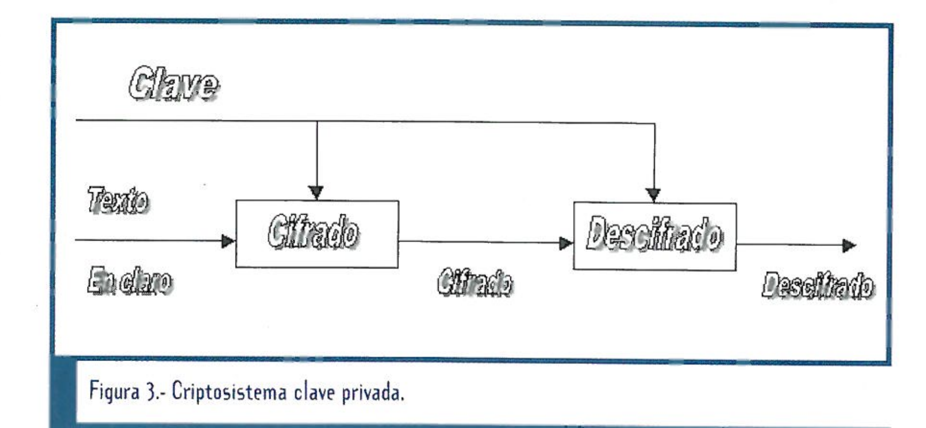


Figura 3.- Criptosistema clave privada.

La implementación de los métodos congruenciales en Java es sencilla y rápida

Aunque usando estos métodos pueden aparecer determinados "huecos" en los ciclos generados, éstos se pueden evitar utilizando un valor de m muy elevado, pero que sea el número primo más grande que se pueda representar.

También se debería utilizar como Z_0 (semilla inicial) un número que fuera primo con m.

Veamos un sencillo ejemplo de cómo generar secuencias de números utilizando *Java*. En el Listado 1 incluido en el *CD-ROM* se encuentra el código fuente del programa generador. Es muy sencillo de entender y apenas consta de complicaciones. Basta con aplicar las fórmulas e intentar utilizar los operadores de exponenciación que se nos presenten adecuadamente en función del ancho de palabra según el sistema en el que estemos desarrollando.

Ahora bien, ¿cuál será el mejor método para seleccionar una buena semilla o valor inicial?. Observe cómo se ha resuelto en el ejemplo. ¿Es totalmente aleatorio?. En la Figura 2 se puede comprobar la ejecución del programa presentado. La secuencia parece verdaderamente aleatoria, y casi cuesta creer que conste de un período de repetición.

3.- Serie de Fibonacci.

Se trata de un método congruencial aditivo, en el que la secuencia de números se determina con:

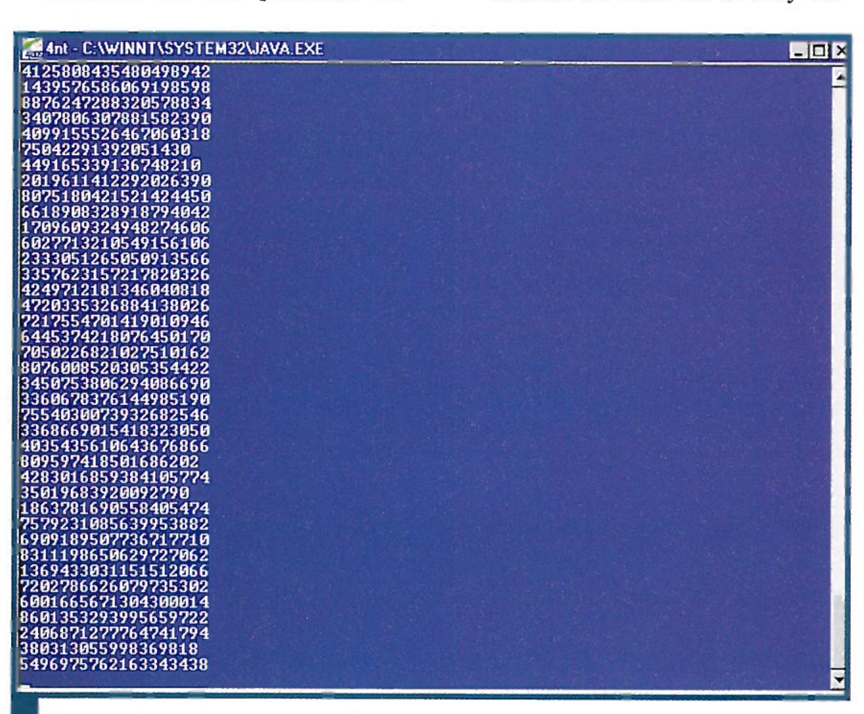


Figura 2.- Secuencia numérica con el método congruencial.



LISTADO 1. Ejemplo del método congruencial multiplicativo

```
import java.io.*;

class congru
{
     PrintStream p = System.out;
     long cte_a  ;
     long cte_m ;

     long valor_ant ;
     long valor_act ;
     congru ( final long l_palabra )
     {
          cte_a = (long) ( Math.pow ( 2, (l_palabra/2) + 1 ) ) + 3
;
          cte_m = Long.MAX_VALUE ;
          valor_ant = valor_act = (long) System.currentTimeMillis();
     }

     long rand ()
     {
          valor_ant = valor_act ;
          valor_act = Math.abs ( cte_a * valor_ant % cte_m ) ;
          return valor_act ;
     }

     public static void main ( String args[] )
     {
          int cont ;

          congru ref_cong = new congru ( 64 );
          for ( cont = 0 ; cont < 1000 ; cont ++ )
          {
               System.out.println ( ref_cong.rand());
          }

     }
}
```

```
Z0  = Z1  = 1,            Zi+2 =
    Zi+1 + Zi (mod m)
```

donde se tiene que *m* se construye de la misma manera que con los anteriores métodos.

Además, nada impide que se puedan construir otros generadores de forma personalizada. La ventaja inmediata de realizar esta tarea es que el algoritmo no es conocido, lo que implica en cierta manera una mayor seguridad, aunque de todas formas no se obtendrá una secuencia puramente aleatoria.

La creación de métodos propios suele pasar por realizar una mezcla de operaciones aritméticas y de congruencia. Por ejemplo, se podría tener el siguiente generador:

```
Paso
a) S= Zi*77317 - 3214516
b) S= mod ( S, 8634729 )
c) Zi+1= s*3216
d) Repetir desde paso a.
```

Uno de los principales inconvenientes de estos sistemas es el elevado tiempo de cálculo que requieren si no son afinados o correctamente diseñados para generar grandes secuencias de números con buenas propiedades estadísticas.

OTROS MEDIOS ALTERNATIVOS DE GENERACIÓN

Estos métodos suelen ser poco utilizados, pero en caso de necesitar una solución rápida a un problema, puede hacerse uso de ellos. Recogemos algunos sencillos de utilizar y aplicar.

1.- Tablas de números aleatorios

En 1955 la empresa *Rand Corporation* publicó un libro que contenía un millón de números aleatorios que cumplían todas las propiedades matemáticas deseables y donde detallaban, con pelos y señales, cómo los habían conseguido, algo bastante complicado dado que la aparición de los ordenadores aún era bastante reciente.

Pregunte en sus librerías favoritas si tienen libros de números aleatorios y en función de la contestación, añada un número entre 0 y 9 a la clave que está generando. Aún así, como es de esperar, no podrá generar números totalmente aleatorios.

2.- Reloj del ordenador

Consiste en utilizar el contenido del registro del reloj del ordenador y, específicamente, unos cuantos *bits*, por regla general los menos significativos. Sin embargo en determinadas circunstancias no debería usarse, en especial si se programa en entornos *UNIX*, (ya que en estos sistemas se utilizan determinadas sincronizaciones).

Este método se empleaba cuando los ordenadores tenían poca potencia, pero en la actualidad, la alta velocidad de los procesadores hace que esta

técnica pueda generar números de escaso valor, ya que no pasan los suficientes "*ticks*" o pulsos de reloj entre una petición y otra.

3.- Latencia de tecleado
La gente suele teclear de forma y con una velocidad más bien aleatoria. Medir el tiempo entre pulsaciones sucesivas y tomar los *bits* menos significativos puede ser un sistema bastante fácil y eficaz de generar secuencias aleatorias.

Hasta hace poco tiempo, la generación mediante el uso de ticks era muy común

Este sistema es muy utilizado en la actualidad, así lo reconocerán todos aquellos usuarios del famoso y excelente producto *PGP*.

CRIPTOSISTEMAS DE CLAVE PRIVADA

Los métodos criptográficos modernos surgen frente a los métodos clásicos porque estos últimos pueden ser atacados computacionalmente con relativa sencillez, ya que la capacidad de cálculo actual es bastante elevada. Por este motivo, su uso queda algo descartado actualmente.

Su conocimiento es importante para saber la base de la criptografía actual y observar la gran evolución de la criptografía moderna.

Los criptosistemas modernos de clave privada o convencionales se caracterizan por utilizar la misma clave tanto para el cifrado como para el descifrado. También se los suele denominar simétricos, en contraposición a los de clave pública o asimétrica que utilizan dos claves diferen-tes para las tareas de cifrado y descifrado.

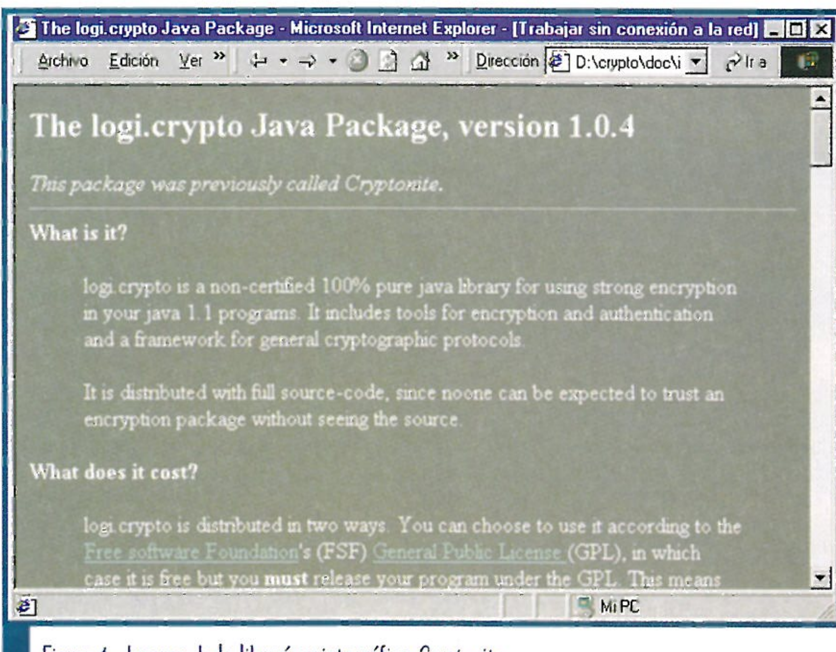


Figura 4.- Imagen de la librería criptográfica *Cryptonite*.

Los métodos clásicos se incluyen también dentro de los criptosistemas de clave privada. Algoritmos como *DES, IDEA, RC5, Blowsfish* pertenecen a este tipo de criptosistemas.

Recordando la información detallada en el anterior artículo, los criptosistemas de clave privada garantizan la confidencialidad ya que se utiliza una clave secreta tanto para cifrar como para descifrar la información. La autenticidad se consigue al permanecer secreta la clave, donde sólo el emisor y el receptor pueden manejar la información, haciendo uso de la clave compartida.

El inconveniente de estos criptosistemas está en la distribución de las claves. Este proceso debe realizarse con todas las garantías de seguridad. Otro inconveniente es el gran número de claves necesarias para intercambiar información entre varios usuarios.

CIFRADO DE PRODUCTO

La gran mayoría de los algoritmos de cifrado simétrico se basan en los conceptos de confusión y difusión propuestos por *Shannon*, que se combinan para dar lugar a los denominados *cifrados de producto*.

Los criptosistemas de clave privada utilizan la misma clave para el cifrado y descifrado

Consiste en ocultar la relación entre el texto cifrado y la clave. El objetivo es alterar las propiedades de la información disponible, principalmente a través de la sustitución.



El propósito de la difusión es distribuir las propiedades estadísticas (redundancia) del mensaje del cifrado. Esto se puede conseguir por ejemplo, con la alteración de la posición de los caracteres (permutación).

La mayoría de los algoritmos se basan en diferentes procesos de sustituciones y permutaciones, estructura que se denomina *Red de Sustitución-Permutación*, como ocurre con el *DES*, que se basa en la repetición de simples sustituciones-permutaciones.

1.- Red de Feistel

Muchos de los cifrados de producto se basan en dividir un bloque de longitud *n* en dos mitades, que se denominan *L* y *R*. Se define entonces un cifrado de producto repetitivo en el que la salida de cada iteración se usa como entrada de la siguiente, según la relación:

```
Li = Ri-1
Ri = Li-1 ≈ f( Ri-1 , Ki) ,  si i
    < n.

Ln = Ln-1 f( Rn-1 , Kn)
Rn = Rn-1
```

Este tipo de estructura se denomina *Red de Feistel* y se emplea en los algoritmos *DES, CAST, Blowfish,* etc. Lo interesante es que es reversible, es decir, el mismo algoritmo se utiliza para cifrar y descifrar.

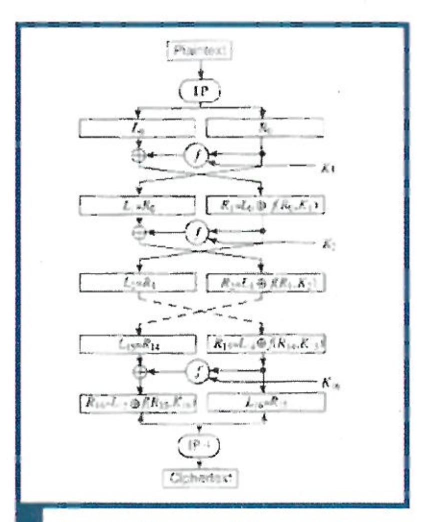


Figura 5.- Algoritmo *DES*.

La mayoría de los criptosistemas de clave privada se basan en el concepto de red de Feistel

2.- Cifrado de Estructura de Grupo

Los cifrados de producto pueden tener estructura de grupo si cumplen la siguiente propiedad:

```
k1, k2, k3 /   Ek1(Ek2(M))=Ek3(M)
```

Esto es, si se hacen dos cifrados con las claves k_1 y k_2, existe una clave k_3 tal que el resultado de aplicar esta última es el mismo que de aplicar las dos anteriores. De ello se deduce que un criptosistema no debería tener esta propiedad.

3.- S-Cajas

Pequeñas tablas utilizadas para aplicar sustituciones (confusión) y permutaciones (difusión).

Una *S-Caja* de *m**n* es una tabla de sustitución que tiene como entrada cadenas de *m bits* y devuelve como salida cadenas de *n bits*.

A continuación se presentan los algoritmos simétricos más comúnmente utilizados. Cada uno posee sus características especiales, tamaño de cifrado del bloque, longitud de la clave, resistencia a posibles ataques, velocidad de utilización, patente vigente, etc.

Se ofrece una breve descripción de su funcionamiento y posteriormente se mostrará cómo utilizar algunos de los más importantes mediante el uso de implementaciones *Java* y de una librería criptográfica muy fácil de uso denominada *Cryptonite (*actualmente denominada *logi. crypto),* gratuita y totalmente desarrollada en *Java*. Se encuentra disponible en el *CD-ROM* que se incluye con la revista.

ALGORITMO DES

Este algoritmo fue desarrollado por *IBM* como continuación al conocido como *Lucifer*. Fue tomado como algoritmo estándar de cifrado por el *NIST (National Institutue of Standars and Technology*) en 1977 para aplicaciones no clasificadas del gobierno de Estados Unidos.

Desde entonces, siempre ha estado sumido en un conjunto de críticas cuyo resultado ha sido concienciar que su uso puede albergar cierta "inseguridad".

El algoritmo cifra bloques de 64 *bits* de texto en claro en bloques de 64 *bits* de texto cifrado. Para ello, se utiliza una clave de 64 *bits* (en realidad son 56 *bits*, ya que los 8 *bits* son de paridad).

El algoritmo es una red de *Reistel* de 16 pasos o iteraciones. En cada una de ellas se utilizan operaciones de "O-exclusivo", permutaciones y sustituciones. Las permutaciones son de tres tipos: simples, expandidas (en las que se duplican *bits*) y restringidas, en las que se eliminan ciertos *bits*. Las sustituciones son del tipo no-lineal, que se expresan en las *S-Cajas*.

Para el cifrado se realizan las siguientes operaciones que se detallarán brevemente.

Se toman los 64 *bits* del bloque de entrada y se realiza una permutación inicial. El bloque resultante B_0, se divide en dos de 32 *bits*, L_0 y R_0. La salida ($L_{16}R_{16}$) después de 16 iteraciones con la función *f*, se permuta de nuevo a la inversa, obteniéndose el cifrado de los 64 *bits* de entrada.

En cada iteración la función *f* combina operaciones de sustitución y transposición. En la Figura 5 se puede observar todo este proceso gráficamente, lo que ayudará a su mejor comprensión. Puede parecer muy enrevesado, pero prácticamente es un proceso repetitivo de tratamiento de *bits*.

LISTADO 2. Ejemplo práctico de la utilización de DES.

```
import is.logi.crypto.*;
import is.logi.crypto.keys.*;

import java.util.Random;
import java.io.*;

public class pdes
{
    String clave ;
    public String texto_claro ;
    String texto_cifrado ;
    String texto_descifrado ;

    byte [] b_claro ;
    byte [] b_cifrado ;

    CipherKey dk ;

    pdes (String pClave,
              String pTextoClaro) {
      clave = pClave;
      texto_claro = pTextoClaro;
      dk = crearClave ();
    }

    CipherKey crearClave () {
      return (CipherKey)
                   new DESKey(clave.getBytes());
    }

     String cifrar () {
        StringBuffer mensaje_aux =
            new StringBuffer ( texto_claro );

        int multiplo_8 = mensaje_aux.length() % 8 ;
        for (int cont_car = 0 ; cont_car < (8-multiplo_8) ; cont_car ++ )
         {
           mensaje_aux.append(" ");
         }

        String mensaje = mensaje_aux.toString();
        b_claro =  mensaje.getBytes();
        b_cifrado = new byte [ mensaje.length() ] ;

        for ( int cont_car = 0 ; cont_car < b_claro.length ; cont_car +=8 )
        {
          dk.encrypt( b_claro, cont_car, b_cifrado, cont_car);
        }

         texto_cifrado =  Crypto.hexString( b_cifrado) ;

         return texto_cifrado;
        }


         String descifrar () {
           for ( int cont_car = 0 ; cont_car < b_cifrado.length ; cont_car +=8 )
         {
           dk.decrypt( b_cifrado, cont_car, b_claro, cont_car);
         }

         texto_descifrado =  new String(b_claro) ;

         return texto_descifrado;
     }

     public static void main(String[] arg)
         throws Exception {

         String cla, tex ;

         DataInputStream di =
              new DataInputStream ( System.in );
         System.out.println ( "Teclee clave (8 car.) : " );
         cla = di.readLine();

         System.out.println ( "Texto a cifrar : " );
         tex = di.readLine();

         pdes prueba = new pdes ( cla, tex );

         System.out.println ( "Texto a cifrar : "  + prueba.texto_claro );

         System.out.println ( "Texto cifrado  : "  + prueba.cifrar() );

         System.out.println ( "Texto descifrado  : "  + prueba.descifrar() );

         System.in.read();
     }
}
```

1.- Variantes del DES

En los entornos *UNIX* se utiliza una variante de *DES* que se denomina *Crypt(3)*, utilizada como una función unidireccional para cifrar las palabras clave de paso.

Otra variante muy utilizada es *TripleDES*, presentada brevemente a continuación, que frente a la inseguridad del *DES* por los escasos *bits* que utiliza para la clave y la posibilidad de un ataque bruto sobre la misma, presenta una clave de mayor longitud, pero a su vez representa todo un proceso mucho más lento (aproximadamente tres veces).

Pero antes de seguir mostrando algoritmos, vamos a utilizar *DES* mediante la librería *Cryptonite,* incluida en el *CD-ROM* de la revista, gracias a la cual el lector comprobará lo sencillo que es utilizar los métodos criptográficos para garantizar la seguridad de los datos en las aplicaciones que desarrolle.

Lo interesante de la librería es su gran sencillez de utilización,



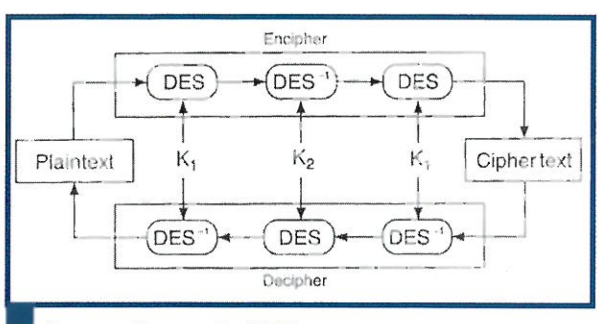


Figura 6.- Algoritmo *TripleDES.*

manejo y aprendizaje. Con una cuantas grandes líneas maestras se pueden utilizar los algoritmos que implementa para realizar la mayoría de las tareas que se puedan presentar.

En el Listado 2 se presenta el código fuente de un programa que cifra una línea de texto introducida por teclado, previa introducción de una clave cualquiera. Este ejemplo puede servir para realizar una aplicación que cifre y descifre un archivo de datos, o las comunicaciones entre un cliente y un servidor a través de una red.

Esos mismos ejercicios se presentarán seguidamente pero aplicando diferentes algoritmos, aunque la filosofía que se va a utilizar es la misma.

Inicialmente necesitamos utilizar las clases propias de la librería *Cryptonite*, por lo que se realiza un *"import"* de **is.logi.crypto.*** y de **is.logi.crypto.keys.***. A partir de este momento, podemos acceder a todas las posibilidades de esta librería.

La clase *pdes* consta de una serie de objetos *String* que almacenarán la clave, el texto en claro introducido, el texto cifrado y el descifrado. Además, se utilizarán dos tablas de *bytes* en los que se realizarán las tareas de cifrado y descifrado.

La primera parte importante aparece con *CipherKey*, que es una interfaz con la que se implementan todas las tareas para realizar el cifrado y descifrado de los datos en función del algoritmo que seleccionemos de los disponibles.

Por tanto, *dk* va a ser una referencia de ese tipo de interfaz y permitirá gestionar un objeto tipo *DESKey*, que es el algoritmo que nos interesa en este ejemplo.

Con llamadas del estilo *dk.encrypt()* y *dk.decrypt()* se realizarán todas las iteraciones de cifrado y descifrado descritas anteriormente sobre la tabla de *bytes* correspondientes.

La utilización de DES resulta relativamente sencilla

En el método *cifrar()* simplemente se ajusta el tamaño de bloque que cifrar por si no tiene el tamaño adecuado y se cifra el *String* utilizado transformándolo anteriormente a su correspondiente tabla de *bytes*. Se repite este proceso tantas veces como bloques de 8 *bytes* se obtengan, llamando por cada uno de ellos a la función *dk.encrypt()*.

Unas funciones importantísimas pertenecientes a *Cryptonite* son *Crypto.hexString()* y *Crypto.fromHexString()*. Con la primera de ellas podemos visualizar el resultado de un cifrado (que se almacena en una tabla de *bytes*), porque devuelve como resultado un objeto de tipo *String* que al visualizarse mostrará secuencia de caracteres hexadecimales, algo visible.

El siguiente método hace el proceso contrario. De un objeto *String* que contiene una secuencia hexadecimal (algo cifrado) se genera una tabla de *bytes* que puede ser utilizada en el proceso de descifrado. Por tanto, devuelve una tabla de *bytes*.

Visto esto, en el método *descifrar()* se realiza el mismo proceso que en *cifrar()*, pero a partir de una tabla de *bytes* que es lo que se descifra e invocando al método *dk.decrypt()* para realizar el descifrado sobre los *bytes*. El resultado es una tabla de *bytes* que, convertida a objeto *String*, visualiza el resultado obtenido.

TRIPLEDES

Se presenta como una variante, una buena mejora del método *DES* clásico. El incremento de la seguridad se realiza mediante un sistema de cifrado-descifrado-cifrado denominado *EDE* (*Encryption-Desencryption-Encryption*) sobre el bloque de texto en claro pudiendo usar dos o tres claves diferentes. Por tanto, las claves ahora son de 112 ó 168 *bits*.

Como habrá sospechado, con esos cálculos rápidos, faltan *bits*. Pues sí, al igual que en *DES*, se desprecian ciertos *bits* para cuestiones de paridad. Formalmente las claves serían de 128 y 192 *bits* respectivamente.

El bloque de datos a cifrar sigue siendo el mismo utilizado en *DES*. Ahora, un ataque por la fuerza bruta llevaría muchísimo más tiempo, ya que el número de *bits* por clave se ha incrementado considerablemente.

Para comenzar el cifrado de un bloque de datos se realizan 16 iteraciones usando la primera clave. A continuación, usando el bloque de datos resultante, se realizan otras 16 iteraciones descifrando esos datos

```
4nt - C:\JDK1.1.7\BIN\JAVA.EXE
[d:\tmp\cripto2]java tripledes
Teclee clave  (24 car. minimo) :
en un lugar de la mancha
Teclee datos ( fin->salir) :
Revista Solo Programadores.

Criptografia con Java

Esto es una prueba de uso de
TripleDes.

Puedo estar tecleando datos
hasta que se detecte "salir"


Vamos a ver el resultado.

salir
Se cree un fichero datos.txt con el texto tecleado cifrado


Descifrando fichero datos.txt cifrado con TripleDES
Revista Solo Programadores.

Criptografia con Java

Esto es una prueba de uso de
TripleDes.

Puedo estar tecleando datos
hasta que se detecte "salir"


Vamos a ver el resultado.

Fin del descifrando fichero datos.txt
```

Figura 7.- Ejemplo de ejecución con *TripleDES*.

mediante el uso de la segunda clave. Finalmente, utilizando la primera clave, se realizan otras 16 iteraciones sobre el resultado de aplicar la segunda clave, resultado el texto cifrado final.

En la Figura 6 se muestra el proceso descrito gráficamente. Como observará, el número de pasos a realizar es mucho mayor, de ahí que sea bastante más lento que *DES*.

En el Listado 3, que se incluye en el *CD-ROM* se muestra cómo utilizar el algoritmo *TripleDES* para cifrar y descifrar el contenido de un archivo de datos que el usuario ha creado utilizando una determinada clave, que únicamente él conoce y no se almacena en ningún otro sitio.

TripleDES es más lento de ejecución que DES

Esto puede ser un añadido que queda como ejercicio. Imagine que sistemas de gestión de claves pueden existir e intente implementar una solución elemental para alguno que haya encontrado.

Aunque el listado 3 se presente como algo más complicado, en realidad, si se utiliza una misma filosofía de aplicación de los sistemas de cifrado común, no presenta ninguna nueva dificultad.

Realizados una serie de *"imports"* necesarios para compilar, que usted mismo podrá ir descubriendo, tenemos la clave **tripledes**. Como en el ejemplo anterior, disponemos de objetos *String* para la clave, texto en claro, cifrado y descifrado. A su vez, de las correspondientes tablas de *bytes* y *CipherKey dk*.

Para utilizar en este caso el algoritmo con *dk*, no tendremos más que asignarle un objeto de tipo *TriDESKey* con la clave que vamos a utilizar, como se observa a continuación:

```
dk = new TriDESKey(clave.getBy-
    tes());
```

Utilizaremos una clave de 192 *bits*, por lo que se necesitarán 24 caracteres.

Los métodos para cifrar y descifrar no varían prácticamente, únicamente necesitamos utilizar en el descifrado una llamada al método *Crypto.fromHexString()*, ya que en el fichero de datos se almacena la secuencia hexadecimal del cifrado.

El método que más ha variado es *main()*. ¿Qué ocurre en él?. Prácticamente nada nuevo si ha utilizado ficheros en *Java*. Se crea un fichero denominado *datos.txt* mediante la inserción de los datos tecleados por el usuario. Se almacena en secuencia hexadecimal para su posterior recuperación. Esto se puede realizar directamente con la copia de los *arrays* de *bytes*, pero siempre es preferible poder ver lo que se está guardando.

Una vez creado el fichero, se utiliza en modo lectura y se descifra, visualizado el contenido del mismo descifrándolo con la misma clave.

Estudie el código, compruebe cómo gracias a la programación orientada a objetos el número de cambios que se deben realizar es mínimo y todo se desarrolla cómodamente. En la Figura 7 se puede apreciar el resultado de ejecutar el Listado 3.

CONCLUSIONES

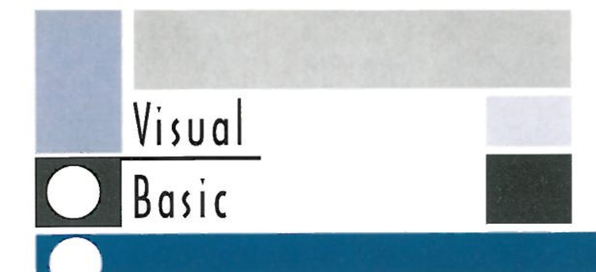
Como se ha podido comprobar a lo largo de este artículo, los medios para generar números aleatorios son muy variados y cada uno presenta sus ventajas y dificultades.

Se han presentado algunos de los criptosistemas privados más comunes. En la siguiente parte se mostrarán otra serie de algoritmos bastante utilizados y se tratarán los criptosistemas de clave pública y el cifrado por bloques.

Con los ejemplos presentados el lector puede comprobar cómo utilizar los algoritmos en sus aplicaciones de forma rápida, eficaz y sencilla, lo que supone un valor añadido en muchas aplicaciones, y en muchas circunstancias algo imprescindible.

BIBLIOGRAFÍA

- **"Applied Cryptography. Protocols, Algorithms and Source Code in C."** Scheneier, Bruce. Wiley Inc.
- **"Security in Computing."** Pfleeger, Charles. Prentice Hall.
- **"Security of Computer based Information Systems."** Lane, V.P. Mac Millan.





Suite para Internet (III)
El correo electrónico (II)

Jordi Agost Moré *(agost@eup.udl.es)*
Profesor de Programación/multimedia en la Universidad de Lleida

En este artículo veremos cómo incorporar un cliente de correo electrónico a nuestra *suite* para que podamos recibir mensajes de servidores. Para ello utilizaremos el protocolo *POP3*.

PROTOCOLO POP3

El protocolo de transmisión de correo electrónico *Pop3* constituye hoy en día, junto a los mensajes de correo electrónico basados en *Web*, la forma más amplia de acceder al correo electrónico disponible en un servidor.

El conjunto de elementos necesarios para incorporar el esquema de *Pop3* se encuentra definido en el documento *rfc2384* fácilmente localizable en *Internet* y consta de un servidor de correo tipo *pop*, un número de puerto a través del cual podamos acceder a dicho servidor, un mecanismo de autenticación, así como un identificador de autenticación y una clave de acceso (*password*).

Una conversación típica entre el cliente y el servidor consta, por lo general, de un intercambio de información similar al siguiente:

- Se pide el *password* al usuario.
- Se efectúa una conexión a un servidor de correo, supuestamente **mail.servidor.com** con el puerto **110.**
- El servidor envía la siguiente respuesta: **+OK POP3 server ready 1986.697170952 @mail.servidor.com.**
- Se produce la respuesta del cliente: **USER rg.**
- A continuación el servidor indica al emisor que ha recibido la información correspondiente al usuario: **+OK.**
- El cliente envía la clave de acceso: **PASS palabra_acceso.**
- Por último, tras validar los datos recibidos, el servidor envía la respuesta: **+OK nombre_cuenta tiene n mensajes (nº octets).**

PROCESO PRINCIPAL

La rutina que iniciará todo el proceso es la que se ejecuta cuando pulsamos el botón de chequeo de correo electrónico. En ella lo primero que hacemos es limpiar el *buffer* de entrada y seguidamente colocamos el identificador de proceso actual a **0** (que corresponde a la identificación del usuario). A continuación y mediante el control *Winsock* nos conectamos con el servidor de correo utilizando el método *Connect*, que recibe como parámetros el nombre del servidor y el puerto que utiliza.

```
Private Sub cmdChequear_Click()
  strBufferEntrada = ""
  intID = 0
  wskPop3.Connect txtServidor.Text,
     CLng(txtPuerto.Text)
End Sub
```

Posteriormente, en el momento en que el control *Winsock* responda se ejecutará de forma automática el evento *DataArrival* del control *Winsock*. Dicho evento será realmente el corazón del sistema, y en él se ejecutará el diálogo de nues-



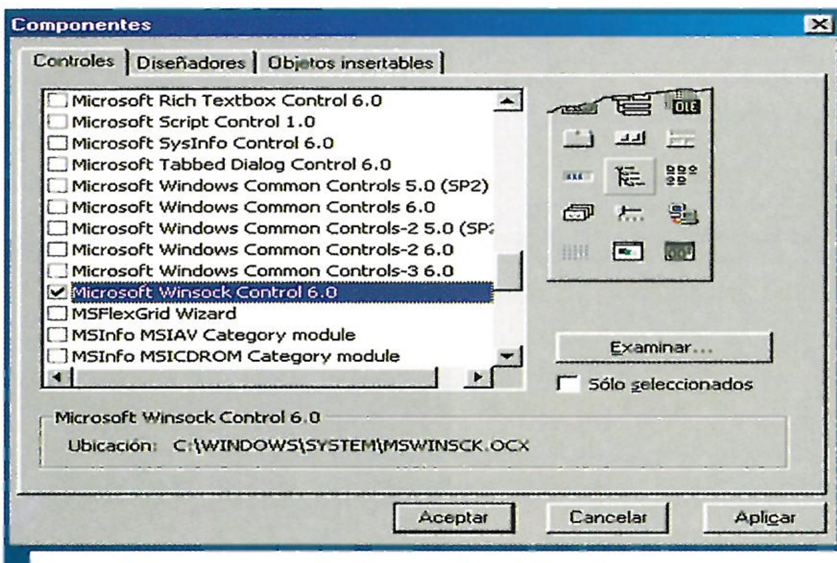


Figura 1.- Tendremos que activar las referencias al control *Winsock*.

tro programa con el servidor. Esta comunicación vendrá establecida por el identificador de proceso, y dependiendo del valor que hayamos establecido para dicho identificador se ejecutará una rutina u otra. Una vez visto el esquema general analizaremos las rutinas de todo el proceso de manera independiente.

Incorporaremos un gestor de correo electrónico para recibir correos de servidores basados en el protocolo POP3

```
Private Sub wskPop3_DataArrival(
  ByVal bytesTotal As Long)
    Select Case intID
        Case 0
            AccionUsuario
        Case 1
            AccionPass
        Case 2
            AccionStat
        Case 3
            AccionList
        Case 4
            AccionRet
        Case Else
                cmdChequear.Enabled =
        True
                wskPop3.Close
    End Select
End Sub
```

PROCESO DEL NOMBRE DE USUARIO

El primer proceso que se ejecutará será el correspondiente a la rutina *AcciónUsuario*, ya que previamente en el botón de chequeo de *mail*, habíamos puesto como identificador de proceso, el número **0**. En dicha rutina lo primero que vamos a hacer va a ser recoger la información enviada por el usuario en una variable llamada **strBufferEntrada2** y unirla con la información que previamente tuviésemos.

A continuación limpiaremos el *buffer* de entrada y pondremos en el de salida el nombre del usuario, para posteriormente enviarlo al servidor con el método *SendData*. Antes de efectuar la acción citada, tendremos que haber aumentado en uno el identificador de proceso, preparando la aplicación para que se ejecute un nuevo proceso cuando se active el evento *DataArrival* del control *Winsock*.

```
Private Sub AccionUsuario()
    wskPop3.GetData strBufferEntrada2
    strBufferEntrada =
        strBufferEntrada &
    strBufferEntrada2
    If Right(strBufferEntrada, 2) =
    CRLF Then
        intID = 1
        strBufferSalida = "USER " &

    txtUsuario.Text & CRLF
        strBufferEntrada = ""
```

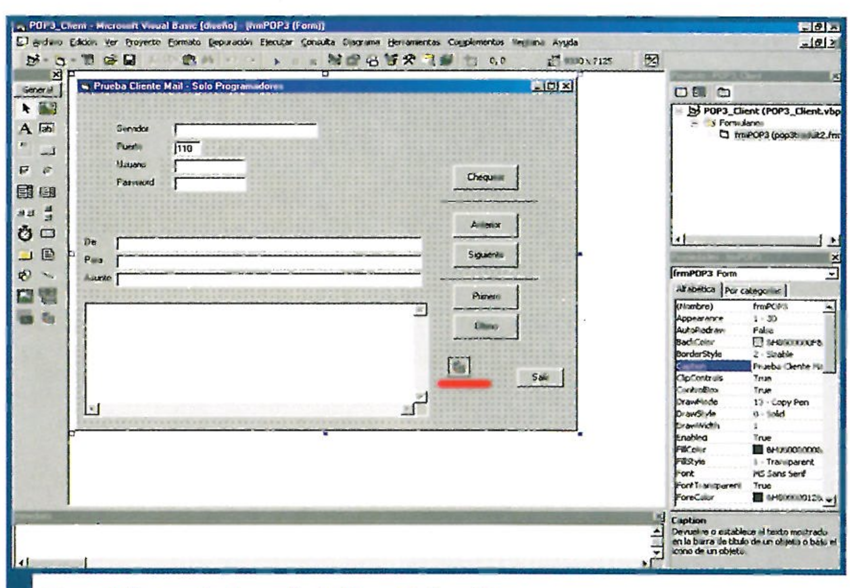


Figura 2.- Entorno de desarrollo de *Microsoft Visual Basic*.

```
        wskPop3.SendData
      strBufferSalida
    End If
End Sub
```

ENVÍO DEL PASSWORD

Cuando el servidor vuelve a enviar datos se ejecutará la rutina *AccionPass*. Al igual que en la rutina anterior, en este caso cogemos los datos del control *Winsock*, los unimos a los que ya teníamos anteriormente, aumentamos el identificador de proceso, y en el *buffer* de salida colocamos el *password* del usuario. A continuación lo enviaremos mediante el control *Winsock*.

Al recibir datos, el control Winsock ejecuta automáticamente el evento DataArrival

Suponiendo que en la transmisión haya existido un error (situación que controlaremos porque en el primer carácter de la respuesta del servidor, si todo ha ido bien, obtendremos el carácter de la suma +), se lo comunicaremos al usuario y cerraremos la transmisión que tenemos abierta.

```
Private Sub AccionPass()
    wskPop3.GetData strBufferEntrada2
    strBufferEntrada =
     strBufferEntrada &
     strBufferEntrada2
    If Right(strBufferEntrada, 2) =
     CRLF Then
        If Left(strBufferEntrada, 1)
     = "+" Then
            intID = 2
            strBufferSalida = "PASS
     " & txt_pass.Text & CRLF
            strBufferEntrada = ""
            wskPop3.SendData
     strBufferSalida
        Else
            MsgBox "Error al enviar
     datos al servidor."
            wskPop3.Close
        End If
    End If

End Sub
```

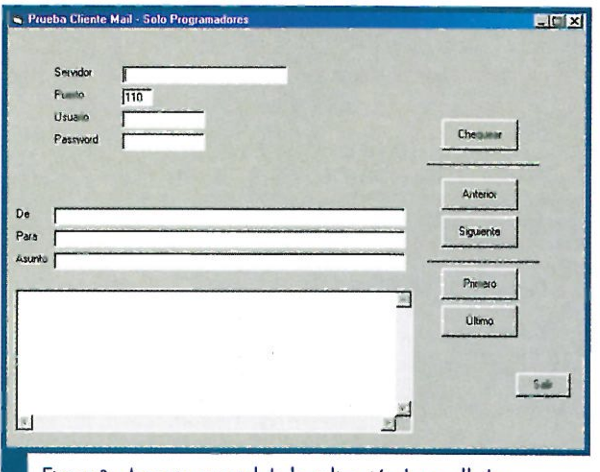


Figura 3.- Aspecto general de la aplicación desarrollada.

INFORME DEL ESTADO

Cuando el servidor vuelva a contestar de nuevo se ejecutará la siguiente rutina:

```
Private Sub AccionStat()
    wskPop3.GetData strBufferEntrada2
    strBufferEntrada =
      strBufferEntrada &
      strBufferEntrada2
    If Right(strBufferEntrada, 2) =
      CRLF Then
     If Left(strBufferEntrada, 1) =
      "+" Then
       intID = 3
       strBufferSalida = "STAT" & CRLF
       strBufferEntrada = ""
       wskPop3.SendData
      strBufferSalida
     Else
       MsgBox "Error al enviar datos
      al servidor."
       wskPop3.Close
     End If
    End If
End Sub
```

En ella observamos cómo se sigue un proceso similar a la anterior. De nuevo diremos que en primer lugar cogemos los datos del *buffer* de entrada, los unimos a los ya existentes, comprobamos si la respuesta del servidor ha sido la correcta, y seguidamente actualizamos el identificador de proceso. Una vez realizado todo el proceso enviamos la orden *STAT* por el *buffer* de salida, limpiando el *buffer* de entrada.

```
Private Sub AccionList()
   wskPop3.GetData strBufferEntrada2
   strBufferEntrada =
     strBufferEntrada &
     strBufferEntrada2
   If Right(strBufferEntrada, 2) =
     CRLF Then
      If Left(strBufferEntrada, 1) =
     "+" Then
         intID = 4
         strBufferSalida = "LIST" &
     CRLF
         strBufferEntrada = ""
         wskPop3.SendData
     strBufferSalida
         intactual = 0
      Else
         MsgBox "Error al enviar
     datos al servidor."
         wskPop3.Close
      End If
   End If
End Sub
```

RECOGIENDO CORREO ELECTRÓNICO

En este punto nos encontramos con la recogida definitiva de los diferentes mensajes de correo electrónico obtenidos. Al entrar en esta parte del módulo deberemos diferenciar dos casos importantes: si es la primera vez que entramos o no. En el primer caso, aún desconocemos el número de mensajes que hay, dato que



Listado 1. Recuperación de los diferentes mails.

```
Private Sub AccionRet()
    wskPop3.GetData strBufferEntrada2
    strBufferEntrada = strBufferEntrada & strBufferEntrada2
If intactual = 0 Then
    If Right(strBufferEntrada, 5) = CRLF_CRLF Then
        intactual = intactual + 1
        If ParserListaCorreos(strBufferEntrada) = 0 Then
            strBufferSalida = "RETR " & intactual & CRLF
            cmdPrimero.Enabled = False
            cmdAnterior.Enabled = False
            cmdSiguiente.Enabled = False
            cmdUltimo.Enabled = False
            cmdIrA.Enabled = False
        Else
            strBufferSalida = "QUIT" & CRLF
            intID = 5
        End If
        strBufferEntrada = ""
        wskPop3.SendData strBufferSalida
    End If
Else
 If intactual < intNumMails Then
    If Right(strBufferEntrada, 5) = CRLF_CRLF Then
        zu = Parse_Mail(strBufferEntrada, intactual)
        intactual = intactual + 1
        strBufferSalida = "RETR " & intactual & CRLF
        strBufferEntrada = ""
        wskPop3.SendData strBufferSalida
    End If
 Else
  If Right(strBufferEntrada, 5) = CRLF_CRLF Then
        zu = Parse_Mail(strBufferEntrada, intactual)
        strBufferSalida = "QUIT" & CRLF
        intID = 5
        strBufferEntrada = ""
        If intNumMails > 1 Then
            cmdPrimero.Enabled = False
            cmdAnterior.Enabled = False
            cmdSiguiente.Enabled = True
            cmdUltimo.Enabled = True
            cmdIrA.Enabled = True
        End If
        lblTotalMails.Caption = "of " & intNumMails
        CargarCampos 1
        wskPop3.SendData strBufferSalida
  End If
 End If
End If

End Sub
```

averiguaremos con la rutina *ParserListaCorreos*.

En dicha rutina contaremos los diferentes *mails* existentes en el *buffer* de entrada (lo sabremos porque están separados por cadenas **CRLF** o el carácter de retorno de carro y de línea juntos) y luego redimensionaremos los diferentes *arrays* donde guardamos la información al valor del número total de *mails* encontrados.

Si no existen mensajes enviaremos la orden *QUIT* al servidor para acabar el proceso, junto con el identificador de proceso número **5**:

```
        strBufferSalida = "QUIT"
& CRLF
        intID = 5
        wskPop3.SendData
strBufferSalida
```

Cada vez que nos desplacemos por los mensajes deberemos habilitar o no los demás botones de desplazamiento

Y en cambio si existen mensajes lo que haremos es pedirle al servidor, mediante la orden *RETR*, que descargue el primero de ellos:

```
    strBufferSalida = "RETR " &
def_mail & CRLF
    wskPop3.SendData
strBufferSalida
```

Mientras realizamos este proceso desconectaremos los botones de desplazamiento a través de los diferentes mensajes para así asegurar la consistencia:

```
        cmdPrimero.Enabled = False
        cmdAnterior.Enabled =
False
```

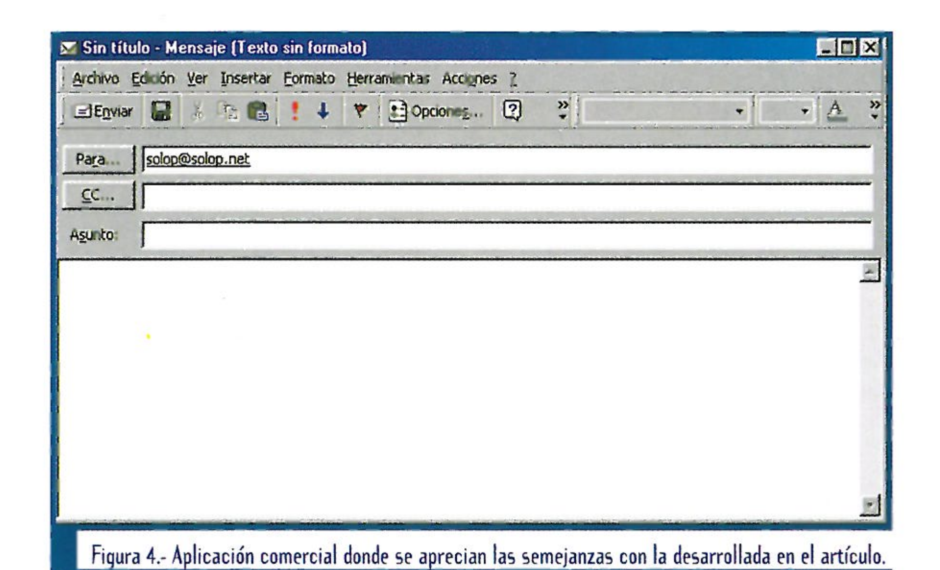


Figura 4.- Aplicación comercial donde se aprecian las semejanzas con la desarrollada en el artículo.

```
        cmdSiguiente.Enabled =
False
        cmdUltimo.Enabled = False
        Cmd_GoTo.Enabled = False
```

Esta parte de la rutina que acabamos de describir tan sólo se ejecutará la primera vez, ya que en las siguientes iremos recogiendo los diferentes mensajes almacenados en el servidor de correo. Lo haremos de manera análoga a como lo hemos hecho para el primero, es decir, a través de la orden *RETR*.

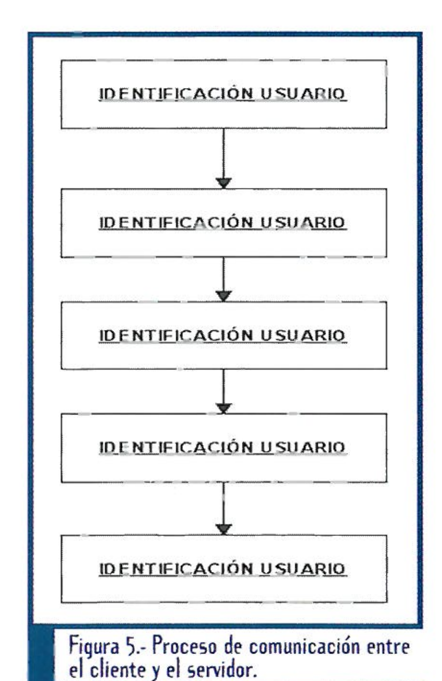


Figura 5.- Proceso de comunicación entre el cliente y el servidor.

En el *CD-ROM* aparece el listado completo de dicha rutina.

NAVEGACIÓN POR LOS MENSAJES RECIBIDOS

Por último deberíamos entender el funcionamiento de las rutinas que nos permiten desplazarnos a través de los distintos correos electrónicos que hemos recibido.

Veremos el código que nos permite desplazarnos al último elemento de los mensajes recibidos. Lo primero que miramos en dicho proceso es si el número que ha escrito el usuario en la caja destinada a tal efecto es realmente un número, entonces si es menor que el número total de mensajes lo igualaremos al número total de correos ya que debemos desplazarnos al último.

Iremos uniendo la información recibida al buffer de entrada ya existente

Seguidamente cargaremos los campos en pantalla y habilitaremos y/o deshabilitaremos los botones correspondientes (el botón **Siguiente** y **Último** han de quedar desactivados ya que nos encontramos en el último elemento). Observemos ahora el código explicado anteriormente:

```
If IsNumeric(txtPosicion.Text)
 Then
    actual =
 CLng(txtPosicion.Text)
    If actual < intNumMails Then
        txtPosicion.Text =
 intNumMails
        txtPosicion.Refresh
        CargarCampos intNumMails
        cmdPrimero.Enabled = True
        cmdAnterior.Enabled = True
        cmdSiguiente.Enabled =
 False
        cmdUltimo.Enabled = False
    End If
End If
```

Miraremos si existe un error comprobando que el primer carácter de la respuesta del servidor sea el de la suma (+)

MIRAR LOS MENSAJES DE LA BANDEJA DE ENTRADA

Vamos a ver ahora una pequeña rutina que puede llegar a ser muy útil para todas aquellas personas que han desarrollado un programa y quieren que éste informe en todo momento al usuario de los nuevos mensajes que van llegando.

Podemos aprovechar aquí algunas de las posibilidades de



Microsoft Office, como por ejemplo la programación de sus diferentes objetos con *Visual Basic* para Aplicaciones, o la posibilidad de acceder a ellos.

Podemos contar el número total de mensajes ya que éstos se encuentran separados por retornos de carro

Para esto último lo primero que deberemos hacer es activar la librería de *Microsoft Outlook* en el apartado de referencias (menú **Proyecto -> Referencias**).

```
Option Explicit
Dim objOut As Outlook.Application
Dim objmap As Outlook.NameSpace
Dim intTmp As Integer
```

Veamos aquí la rutina:

```
Private Sub Form_Click()

Dim I

Set objOut = New Outlook.Application
Set objmap =
     objOut.GetNamespace("MAPI")

For I = 1 To
     objmap.GetDefaultFolder(olFolderI
     nbox).UnReadItemCount
intTmp = intTmp + 1
Next

MsgBox "Tienes " & intTmp & " mensa-
     jes nuevos en la bandeja de
     entrada ", vbInformation +
     vbOKOnly, "ATENCIÓN"

Set objmap = Nothing
Set objOut = Nothing

End Sub
```

Componentes

Controles | Diseñadores | Objetos insertables

- Microsoft Internet Transfer Control 6.0
- Microsoft MAPI Controls 6.0
- Microsoft Masked Edit Control 6.0
- Microsoft Multimedia Control 6.0
- Microsoft NetShow Player
- Microsoft Office Web Components 9.0
- [x] Microsoft Outlook Express 5.0 Type Library
- Microsoft PictureClip Control 6.0
- Microsoft RemoteData Control 6.0
- Microsoft Rich Textbox Control 6.0
- Microsoft Script Control 1.0
- Microsoft SysInfo Control 6.0
- Microsoft Tabbed Dialog Control 6.0

Examinar...

Sólo seleccionados

Microsoft Outlook Express 5.0 Type Library

Ubicación: C:\...\ARCHI..\OUTLOOK EXPRESS\MSOE.DLL

Aceptar | Cancelar | Aplicar

Figura 5.-Una vez realizado este paso, podemos declarar los elementos que nos interesan para acceder a los distintos objetos internos de *Microsoft Outlook*. Esto lo realizaremos en la sección de declaraciones del formulario o módulo en el que estemos trabajando.

Vemos que lo primero que tenemos que hacer en nuestra rutina es asignar un valor a las distintas variables que hemos creado previamente en la sección de declaraciones:

```
Set objOut = New Outlook.Application
Set objmap =
     objOut.GetNamespace("MAPI")
```

Para acceder a los distintos objetos de las aplicaciones de Office, antes deberemos activar las referencias a sus librerías

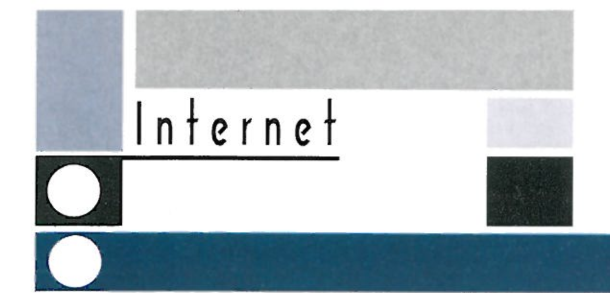
Observemos que a la primera variable se le asigna el valor de una nueva aplicación de *Microsoft Outlook*, y que mediante la segunda accederemos a los objetos de la sección de mensajería de esta aplicación de *Microsoft Office*.

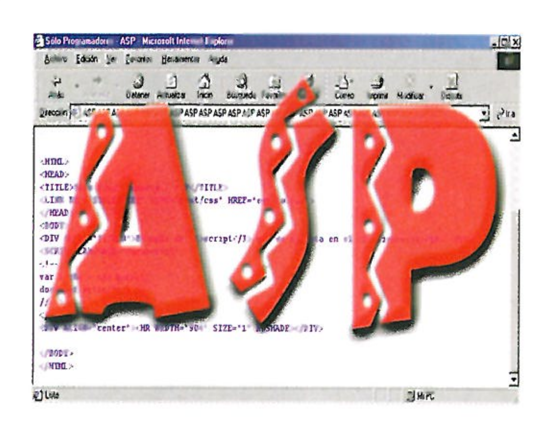
Seguidamente con el método *UnReadItemCount* del objeto referenciado por *objmap* obtendremos el número de elementos de nuestra bandeja de entrada que no hemos leído. Pasaremos este valor a la variable **intTmp**. Después mostraremos la información por pantalla mediante una caja de texto y para finalizar con todas estas operaciones eliminaremos la referencia a las variables creadas. Como éstas son de tipo objeto, dicho paso lo deberemos hacer utilizando la palabra clave *Nothing*:

```
Set objmap = Nothing
Set objOut = Nothing
```

CONCLUSIÓN

En este artículo se ha tratado la forma de crear la base de nuestro propio programa de gestión de correo electrónico. Sin duda, se trata de una buena base para integrar en cualquier aplicación la posibilidad de manejar correo electrónico.





La tecnología ASP (II)
El lenguaje JScript

Adolfo Aladro García. Analista programador

En el capítulo anterior se describieron los fundamentos de la programación de páginas *ASP.* En éste se analizará el lenguaje *JScript,* que permite elaborar programas donde se requiera un nivel de complejidad más elevado.

EL LENGUAJE JSCRIPT

Hasta hace poco los lenguajes de *script* eran considerados como menores, sin embargo con cada nueva versión, han ido adquiriendo potencia y versatilidad, con la ventaja añadida de que han conservado su sencillez. Hoy *JScript* proporciona recursos de programación más que suficientes para llevar a cabo tareas complejas.

JScript es un lenguaje muy potente y versátil

La programación de páginas *ASP* se rige entorno al lenguaje *JScript,* o *VBScript.* Si se pretende llegar a un cierto dominio en el desarrollo de aplicaciones *Web,* es preciso realizar un estudio en profundidad de las características que ofrece. Para aquellos que ya hayan realizado *scripts* de cliente se trata de ahondar en esos conocimientos. Aquellos otros, para los que ésta es la primera vez que se tropiezan con *JScript,* descubrirán que se trata de un lenguaje poderoso y fácil de aprender.

VARIABLES Y TIPOS DE DATOS

La palabra reservada *var* se utiliza para declarar variables. Una de las características que puede llamar más la atención de *JScript* es que es un lenguaje que está escasamente *tipado*. Esto significa que las variables cuando son declaradas no se asocian a ningún tipo de datos concreto. Esta característica, que puede dar lugar a algunos errores al principio, se traduce en una mayor flexibilidad a la hora de desarrollar los *scripts*.

OPERADORES

Veamos algunos de los operadores más importantes de *Jscript*:

- Asignación (=): asigna el valor de la expresión derecha a la variable de la izquierda.
- Operadores artiméticos son: + (suma), - (resta), * (multiplicación), / (división) y % (módulo). El símbolo + también se utiliza como operador de concatenación de cadenas de textos.
- Incrementación (++) y decrementación (--): Suman o restan 1 a su argumento.
- Asignación relativa (+=, -=, *=, /=, %=): Además del operador de asignación, =, existen otros de asignación relativa. Así, el operador += en *i += 3* equivale a *i = i + 3.* Este operador puede utilizarse también con cadenas. Así s += "Lala"



equivale a concatenar **Lala** al final de la cadena contenida en la variable **s.**

- Relacionales: Devuelven *true* o *false* dependiendo del resultado de la comparación: == (igual a), != (no igual a), > (mayor que), < (menor que), >= (mayor o igual que), <= (menor o igual que). Además de los anteriores *JScript* proporciona los operadores de identidad === (idéntico a) y !== (no idéntico a). Es decir, no se realiza conversión de tipos durante el proceso de comparación.

- Lógicos: El operador lógico ! (no) produce *false* si su operando es *true* y viceversa. El operador lógico && (y) produce *true* sólo si ambos operandos son *true.* Finalmente el operador lógico | | (o) produce *true* si cualquiera de los dos operandos es *true.*

- *New:* Crear instancias de objetos. Así, cuando hacemos:

```
var f = new Date();
```

La variable **f** toma una nueva fecha. La expresión que aparece a la derecha del operador *new* tiene que corresponderse con una función constructor de objeto.

- *instanceof:* Devuelve *true* si el operando es una instancia de un determinado tipo de objeto. Así la expresión *f instanceof Date* serviría para saber si la variable **f** contiene un objeto del tipo *Date.*

- *typeof:* Devuelve una cadena de texto que describe el tipo de objeto del operando. Existen seis valores posibles: *number, string, boolean, object, function,* y *undefined.*

SENTENCIAS DE CONTROL

SENTENCIAS CONDICIONALES

- *if...else:* Es la sentencia condicional por excelencia en casi todos los lenguajes.

```
if (i == 1) {
  // Cuando i vale 1 ...
} else {
  // En cualquier otro caso ...
```

Es posible anidar tantos *if...else* como se desee e incluso se pueden utilizar los *if...else if* cuando hay selecciones múltiples.

```
if (i == 1) {
  // Cuando i vale 1 ...
} else if (i == 2) {
  // Cuando i vale 2 ...
} else {
  // En cualquier otro caso ...
```

- *switch:* Constituye otra forma de realizar selecciones múltiples como:

```
switch(i) {
  case 1: // Cuando i vale 1 ...
      break;
  case 2: // Cuando i vale 2 ...
      break;
  default: // En cualquier otro caso ...
      break;
}
```

BUCLES

- *while:* Las sentencias se ejecutan repetidamente hasta que el valor de la expresión se hace *false:*

```
var i = 0;
while (i<3) {
  // Efectuamos algún proceso
  i++;
}
```

- *do...while:* Las sentencias se ejecutan repetidamente hasta que se cumple el valor de la expresión asociada a *while:*

```
var i = 0;
do {
  // Sentencias
  i++;
} while (i<3);
```

La diferencia con respecto al bucle *while* es que en este caso sabemos que las sentencias se van a ejecutan al menos una vez.

- *for:* Tiene el siguiente formato:

```
for (exp1; exp2; exp3) {
  // Sentencias
}
```

Las sentencias se ejecutan repetidamente hasta que el valor de la expresión *exp2* sea *false; exp1* es una expresión de inicialización, y sólo se evalúa una vez al comenzar el bucle; *exp3* se evalúa al final de las sentencias, y en general sirve para modificar el contador del bucle.

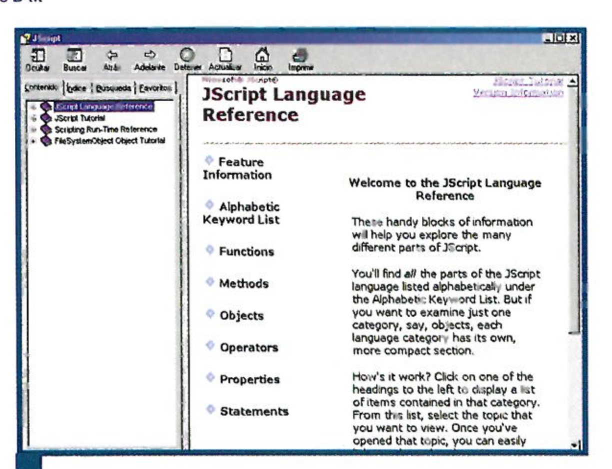


Figura 1.- La documentación de *JScript* se puede descargar del sitio *Web* de *Microsoft.*

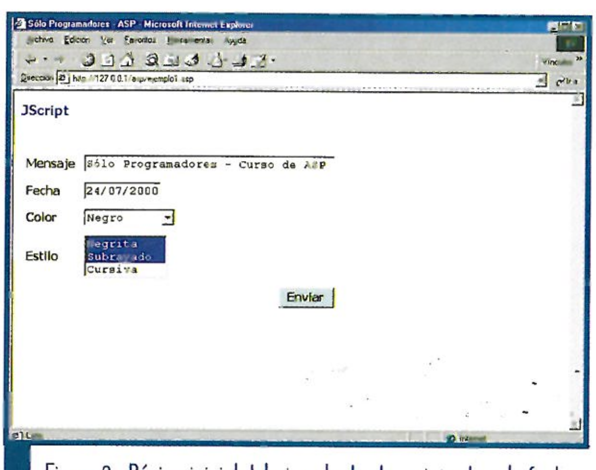


Figura 2.- Página inicial del ejemplo donde se introduce la fecha, el mensaje y el estilo.

```
for (i=0; i<3; i++) {
// Sentencias
}
```

- *break* y *continue:* La sentencia *break* permite interrumpir la ejecución de un bucle. Por su parte *continue* se utiliza para terminar una iteración y pasar a la siguiente.

```
var encontrado = false;
for (i=0; i<cadena.length; i++) {
if (cadena.charAt(i) == "z") {
encontrado = true;
break;
}
}
```

Tras ejecutar el bloque de código anterior la variable encontrada tendrá el valor *true* en el caso de que la cadena tenga al menos un carácter *z* o *false* en cualquier otro caso. Como sólo nos interesa saber si hay al menos una ocurrencia del carácter *z*, en el momento que detectemos una podemos detener la ejecución del bucle *break*.

FUNCIONES

JScript puede crear nuestras propias funciones mediante la utilización de la palabra reservada *function*. Aunque hablamos de funciones en realidad *JScript* no distingue entre procedimientos (funciones que no devuelven valor alguno) y funciones propiamente dichas. Nuestras funciones actuarán de un modo u otro dependiendo de si utilizamos la palabra reservada *return* para devolver un valor.

CONTROL DE ERRORES

Recientemente *JScript* ha incorporado un mecanismo de control de excepciones inspirado en el que tiene *Java*.

```
try {
// Código susceptible de generar errores
} catch (e) {
// Tratamiento del error
}
```

El bloque de código asociado a la sentencia *try* es el que puede generar una excepción. La sentencia *catch* contiene el código que la gestiona en caso de ocurrir dicha excepción. La variable **e**, que puede tomar cualquier nombre, contiene el objeto característico de esa excepción.

Para Jscript funciones y procedimientos son lo mismo

Las excepciones pueden ser lanzadas por el propio sistema o bien por el programador cuando se den las condiciones. En el primer caso hablamos de errores del tipo *run-time-errors* y son aquellos que se producen cuando, por ejemplo, intentamos llamar a un método de un objeto que no existe. Cada vez que se produce un error de este tipo se instancia un objeto predefinido del tipo *Error*. Este objeto cuenta con dos propiedades: *number*, que contiene un número asociado con ese error, y *description*, que contiene la descripción del error. El Listado 1 contiene un ejemplo relacionado con este aspecto. Como un objeto tipo *array* no tiene el método *test* se lanzará una excepción.

El segundo tipo de excepciones son aquellas que el programador lanza. Para ello existe la sentencia *throw*. El único argumento con el que cuenta esta sentencia es la excepción, que puede ser cualquiera (una cadena de texto, un número o incluso un objeto).

OBJETOS, MÉTODOS Y PROPIEDADES

JScript hereda muchas de sus características de lenguajes como *C*, y más directamente de *Java*. De este último toma una "cierta" orientación a objetos, aunque está limitada. No en vano en los *scripts* de cliente los elementos *HTML* se manejan con *JScript* como objetos.

En las páginas *ASP* no tiene sentido hablar de ninguno de estos objetos ya que están vinculados con el navegador, y las páginas *ASP* se ejecutan en el servidor. Sin embargo existen otros objetos que sí podemos utilizar en nuestros *scripts*. Las cadenas de texto y los *arrays* son de los más útiles y frecuentes.

CADENAS

Las cadenas de texto son uno de los elementos más utilizados a la hora de realizar cualquier tipo de programa. *JScript* trabaja con las cadenas de manera similar a como lo hace *Java* (de hecho muchas de las pro-



piedades y métodos del objeto *string* en *JScript* son idénticas). Tal y como dijimos el operador + se utiliza para concatenar cadenas.

```
var s1 = "Express";
var s2 = "Yourself";
var s = s1 + " " + s2;
```

Después de ejecutar el código anterior la variable **s** tiene un valor igual a **Express Yourself**. Las cadenas de texto pueden delimitarse con comillas dobles o simples. Las siguientes declaraciones son equivalentes, por lo que no hay diferencia:

```
var s1 = "Express";
var s2 = 'Express';
```

¿Qué ocurre si queremos representar una cadena que está formada por alguno de esos dos caracteres? Existen dos posibilidades: en primer lugar se puede utilizar el carácter de escape, la barra \, o bien es posible sacar partido al hecho de que existen dos caracteres distintos para delimitar cadenas. La última posibilidad será utilizada en la mayoría de los casos ya que evita tener que estar añadiendo caracteres de escape.

```
var s = 'El pequeño \'Tim\' espera el correo';
var s = "El pequeño 'Tim' espera el correo";
```

Las dos expresiones anteriores son equivalentes y dan como resultado que la variable **s** guarde **El pequeño 'Tim' espera el correo.**

Veamos otro ejemplo:

```
var s = "El pequeño
      \"Tim\" espera el
      correo";
var s = 'El pequeño "Tim"
      espera el correo';
```

En este caso la cadena guardada será **El pequeño "Tim" espera el correo.**

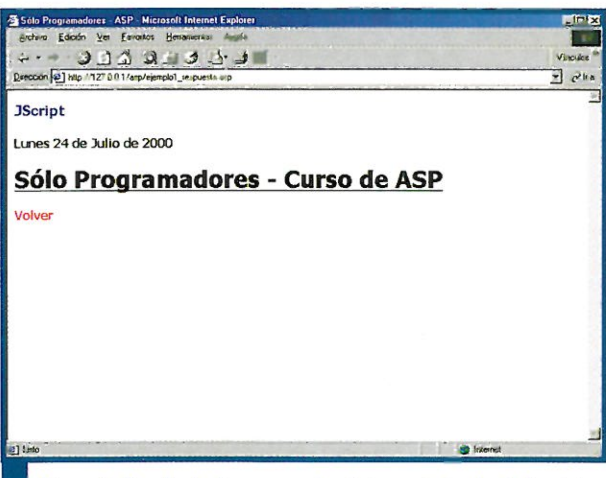


Figura 3.- Resultado de procesar los datos enviados desde la página *ejemplo1.asp*

ARRAYS

JScript proporciona una manera flexible, sencilla y potente de manejar *arrays* de elementos. Podemos asignar valores y/o consultarlos sin preocuparnos de la memoria (ni para reservarla ni para liberarla). El sistema se encarga de todos esos aspectos. Hay varias maneras de trabajar con *arrays*. Veamos cómo podemos declarar un *array* con unos ejemplos:

```
a = new Array();
a = new Array(10);
a = new Array("Frozen", "Swim");
a = new Array(10, 3242);
```

En el primer caso estamos creando un *array* cuyas dimensiones y tamaño se desconocen. Este tipo de declaraciones se utilizan cuando el *array* va a rellenarse dinámicamente a lo largo del *script*.

La segunda declaración sirve para crear un *array* de diez posiciones en el que, de momento, no se almacenan datos. El hecho de que se haya definido el tamaño no es óbice para que más tarde se pueda incrementar. Por otro lado, los elementos de cada posición no tienen por qué ser del mismo tipo. Así por ejemplo:

```
a = new Array(10);
a[0] = "Rozen";
a[1] = 10;
a[2] = new Date();
```

En este ejemplo se ha declarado un *array* de diez posiciones y las tres primeras han sido inicializadas con una cadena de texto, un número entero y una fecha. El resto permanece sin inicializar. La posibilidad de poder asignar cualquier tipo de datos es muy útil cuando queremos crear colecciones complejas de objetos.

Los dos últimos casos muestran sendos ejemplos de *arrays* que se crean ya con una serie de valores inicializados. El primero representa un *array* de cinco cadenas de texto y el segundo un *array* de cinco números enteros.

Estos objetos cuentan con numerosas propiedades y métodos que iremos conociendo. La propiedad que usaremos más a menudo es *lengh*, que devuelve el tamaño del *array*. Nótese que las posiciones de

LISTADO 1. Utilización del objeto error para excepciones como *run-time error*.

```
try {
 x = new Array(1,2,3);
 x.test();
 } catch (e) {
    resultado = 'e = ' + e + '<BR>';
   resultado += 'e.number = ' + e.number + '<BR>';
   resultado += 'e.description = ' + e.description + '\n';
   Response.Write(resultado);
 }
```

Las fechas, cadenas de texto, etc., se tratan como objetos

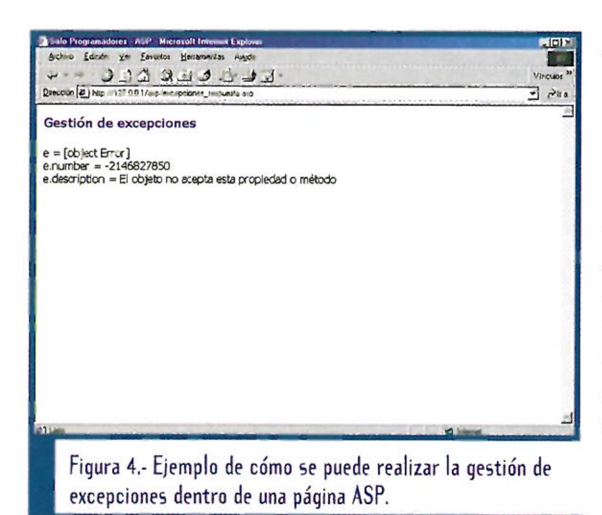


Figura 4.- Ejemplo de cómo se puede realizar la gestión de excepciones dentro de una página ASP.

un *array* se numeran del **0** en adelante. Así por ejemplo, una excelente manera de crear dinámicamente elementos en un *array* es:

```
a[a.length] = 3;
```

Con esta expresión se introduce el número tres al final del *array*.

EL EJEMPLO

La página **ejemplo1.asp,** en el *CD-ROM* contiene un formulario constituido por cuatro elementos: una caja de texto donde se introduce un mensaje y otra que recoge una fecha con formato *DD/MM/YYYY* (día y mes con dos cifras y año con cuatro); y finalmente dos listas desplegables. La primera selecciona un color y la segunda ofrece la posibilidad de elegir varios estilos: negrita, cursiva, subrayado, cualquier combinación o ninguno.

Nuestra página *ASP* receptora realizará dos tareas fundamentalmente. Por un lado tomará la fecha enviada desde el formulario y mostrará un mensaje con la fecha en formato extendido. Por ejemplo: **Sábado 18 de Septiembre de 1999**. En segundo lugar, el mensaje aparecerá en la página con el color y estilo indicados.

Para la primera utilizaríamos tres objetos que proporciona *JScript*: *string*, *date* y *array*. El objeto *date* sirve para crear fechas y cuenta con numerosos métodos con los que es posible obtener el día de la semana que corresponde a esa fecha, el mes, etc. Para crear una fecha es necesario tener separados los valores correspondientes al día, mes y año, por ejemplo **18/09/1999.**

El tamaño de un array puede modificarse dinámicamente

El método *substring* de un objeto *string*, es decir, de una cadena de texto, devuelve la subcadena que se forma tomando los caracteres de la cadena original que se encuentran entre una posición inicial y otra final.

```
0 1 2 3 4 5 6 7 8 9
1 8 / 0 9 / 1 9 9 9
```

La función *parseInt* convierte una cadena de texto en un número. Utilizándola nos aseguramos que, por ejemplo, el mes **09** se almacene en la variable como el mes **9**.

```
dia = parseInt(cadena_fecha.substring(0,2),10);
mes = parseInt(cadena_fecha.substring(3,5),10) - 1;
ano = parseInt(cadena_fecha.substring(6,10),10);
fecha = new Date(ano, mes, dia);
```

El motivo por el que en nuestro ejemplo se resta una unidad al valor del mes se debe a que el objeto *Date* contabiliza los meses del **0** al **11**.

El método *getDay* de un objeto devuelve un valor número correspondiente al día de la semana. La semana inglesa comienza en domingo pero nosotros queremos mostrar no el número sino el nombre del día de la semana, por lo que establecerás:

```
var dias_semana = new Array("Domingo",
        "Lunes", "Martes",
"Miércoles", "Jueves", "Viernes", "Sábado");
```

En el *array* los días y los días están ordenados según la semana inglesa. Como los elementos de un *array* se contabilizan empezando en **0** el valor devuelto por el método *getMonth*, que será entre **0** y **11**, nos llevará directamente al nombre del mes buscado.

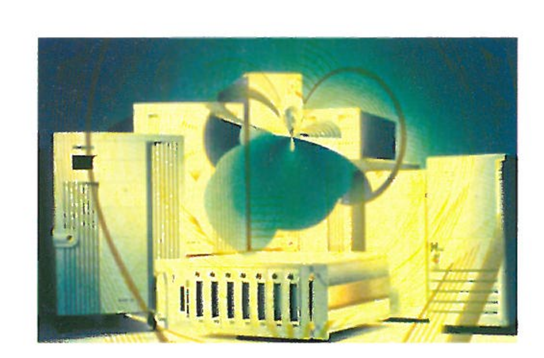
Finalmente la cadena que muestra la fecha queda como sigue:

```
<DIV><B><%= dias_semana[fecha.getDay()]
        + " " + dia + " de " +
meses[mes] + " de " + ano
        %></B><BR> </DIV>
```

La segunda de las tareas consiste en generar el código *HTML* preciso para mostrar el mensaje recibido tal y como indican el color y estilo. Para ello se construirá una cadena de texto con código *HTML* en función de los parámetros recibidos. De entre todos el más problemático es el estilo ya que puede que no llegue ningún valor, uno o varios separados por comas. Para detectar la presencia de cualquiera de ellos utilizaremos el método *indexOf* del objeto *string*.

CONCLUSIÓN

En la próxima entrega profundizaremos en las aplicaciones *Web* avanzadas. Sin embargo, para ir abriendo boca os aconsejamos visitar la dirección: **http://msdn.microsoft.com/scripting/**



Desarrollo cliente/servidor (y V)

Javier Toledo. Analista Programador

Hasta ahora se han expuesto las características de una aplicación cliente/servidor. Para finalizar esta serie mostramos los pasos para la creación de la base de datos en *SQL Server* y las modificaciones en nuestro programa.

CREACIÓN DE BBDD EN SQL SERVER

Para crear nuestra base de datos *COMERCIOS* en *SQL Server 7.0* debemos utilizar la herramienta *Enterprise Manager* y conectarnos a nuestro servidor *SQL Server* como un usuario con derechos de creación de bases de datos (por ejemplo **sa**).

Una vez conectados procederemos a su creación. Para ello nos situaremos sobre la rama del árbol *Databases* que cuelga de nuestro servidor. Pulsando con el botón derecho del ratón tendremos accesible la opción *Nueva base de datos*, desde la que se abrirá una ventana para especificar la ruta a los ficheros de nuestra base de datos y nombre.

La ruta nos es indiferente y puede ser seleccionada a conveniencia del administrador. Pero el nombre de la base de datos deberá ser **COMERCIOS**. Después procederemos a crear nuestro usuario y las tablas de nuestra aplicación.

Para ello deberemos abrir la utilidad *SQL Server Query Analizer* desde el *Enterprise Manager*, con el fin de ejecutar *scripts* contra una base de datos. Una vez seleccionada la base de datos *COMERCIOS* en el desplegable mostrado en la parte superior podremos lanzar el *script* desde esta aplicación, tal y como se puede observar en el Listado 1, incluido en el *CD-ROM*.

Llegados a este punto ya deberíamos tener creada nuestra base de datos con todas sus tablas. Para verificar que existen las relaciones entre las tablas podemos visualizarlas al crear un nuevo diagrama.

Antes de poder utilizar esta base de datos en nuestra aplicación deberemos establecer una conexión *ODBC* con nuestro servidor de datos. Para ello debemos tener instalados los *drivers* convenientes en nuestra máquina cliente.

ODBC

Un consorcio de compañías de *software* lideradas por *Microsoft* fueron los creadores de la tecnología *ODBC*. *ODBC* es una *API*, con la que accedemos a nuestros gestores de bases de datos. Cada sistema de gestión (*SGBD*) tiene sus *drivers ODBC* personalizados.

Todas estas tecnologías tienen en común una serie de características. La finalidad de las mismas es facilitarnos la tarea de programar contra una base de datos, utilizando sentencias *SQL* estándar. Con lo que nos olvidamos de las particularidades de cada gestor, lo que implica que la migración de un *SGBD* a otro no afectará a nuestras aplicaciones. En definitiva, esta tecnología permite desarrollar aplicaciones independientemente del protocolo utilizado. Los elementos que la forman son:

- Una manera estándar de conectarse a un *SGBD* para acceder a sus datos.



- Una sintaxis *SQL* basada en las especificaciones de *X/Open* y el *SQL* del *SAG* (1992).
- Un conjunto estándar de tipos de datos.
- Biblioteca de funciones. Un conjunto de funciones *ODBC* mediante las cuales se permite el acceso a los datos de los diferentes *SGBD*.
- Códigos estándar de errores para su correcta interpretación por la aplicación.

Los beneficios son:

- Las sentencias *SQL* pueden incluirse en el código o construirse en tiempo de ejecución.
- El código de la aplicación es portable a cualquier *SGBD*.
- La aplicación es independiente del protocolo de red existente.

LA CONEXIÓN ODBC

Para crear una fuente de datos *ODBC* debemos acudir al *Panel de control* y seleccionar el icono *ODBC*. Se abrirá un cuadro de diálogo desde el que podremos escoger el tipo de fuente de datos que deseemos crear. Para ello pulsaremos sobre el botón **Nuevo** y seleccionaremos un *driver* para *SQL Server*. A continuación iremos introduciendo todos los datos necesarios:

- *Nombre*: **ODBCCOMERCIOSSQL.** Nombre por el que nos referiremos a la conexión.
- *Server*: ruta a nuestro servidor de *BBDD SQL Server*.
- *Tipo de autenticación*: Autenticación *SQL*.
- *Usuario*: **comercios.**
- *Password*: Aunque nuestro usuario no tiene podemos establecerla.
- *Base de datos por defecto*: **COMERCIOS.**

En el caso de *Oracle* los pasos son semejantes, pero deberemos configurar previamente una conexión con la herramienta de *Oracle SQL Net easy configuration*. Esta herramienta configura una conexión a más bajo nivel que *ODBC*, que será utilizada por éste para efectuar su conexión. En el listado de la aplicación se ha de modificar la cadena de conexión del objeto *Connection*, quedando como se muestra en el Listado 2, incluido en el *CD-ROM*.

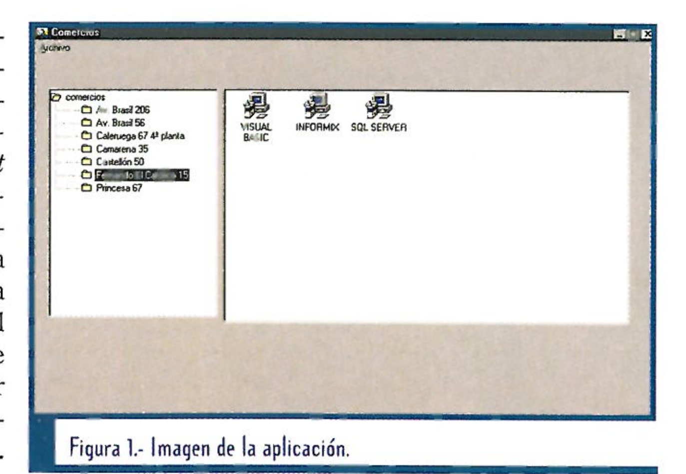


Figura 1.- Imagen de la aplicación.

Antes de ejecutar nuestra aplicación en el caso de que la base de datos sea *Oracle* deberemos cambiar las sentencias *SQL* que *Oracle* no entiende (los *innerjoin*. *Oracle* entiende los *join* de *SQL* de una manera semejante aunque no idéntica). Además debemos llenar nuestras tablas de productos, comercios y comercios-productos con datos antes de arrancar nuestra aplicación con *SQL SERVER*. Podemos copiarlos de **comercios.mdb**. Finalmente debemos comprobar que los ficheros que nos hemos copiado del *CD-ROM* tengan permiso de escritura.

CARACTERÍSTICAS ORACLE 8

OPTIMIZADOR SQL

Nuestro servidor de bases de datos es el encargado de responder nuestras peticiones *SQL* en el menor tiempo posible.

Supongamos una *select* que involucre diez tablas distintas. Evidentemente para resolver esta consulta el gestor de bases de datos tiene que ir cogiendo las tablas una a una e ir mezclándolas en función de la cláusula *where* utilizada. Pero, ¿cuál es el orden óptimo para ir mezclando las tablas? ¿debe escoger primero las tablas con menos datos? ¿debe escoger primero las tablas que crea en función del tipo de mezcla *inner join* antes de *outer join*, etc.? Existen diversas políticas para discriminar estas decisiones:

- Optimización independiente de la sintaxis basada en costes.
- Generación de estadísticas, *ANALYZE*.
- Selección de bucle anidado y fusión *sort-merge*.
- Soporte de consultas multidimensionales "Esquema en Estrella".

ACCESO A DATOS

Con el fin de acelerar el acceso a los registros de nuestras tablas, *Oracle* dispone de diversas posibilidades que pueden sorprendernos por su versatilidad. Diversos tipos de índices que habremos de elegir teniendo en cuenta la cardinalidad de nuestros datos así como el tipo de operaciones que realizaremos sobre ellos (*select*, *update*, etc.). Almacenamiento de dos tablas conjuntas dado que siempre se consultarán en *inner join*, etc.

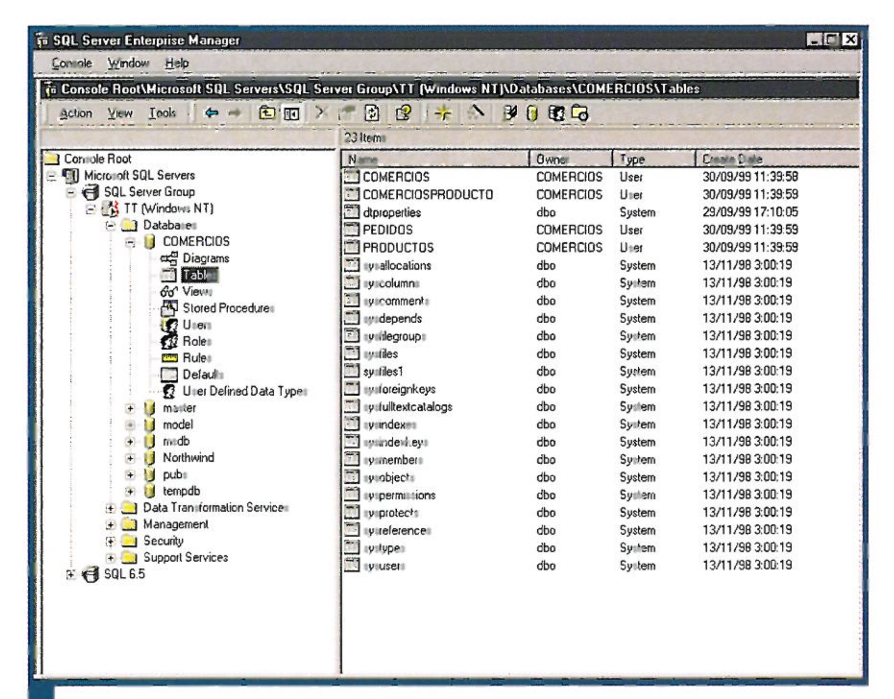


Figura 2.- Base de datos de Comercios en *SQL Server*.

- Indices *B-tree*, monocolumna y de columna concatenada.
- Tablas por *cluster*, *hash-clusters*.
- *ROWID*.
- Resultados de consulta directamente a través de búsqueda en el índice.
- Soporte de índices *bit-map*.

CONTROL DE CONCURRENCIA Y RESULTADOS FIABLES

- Bloqueo al nivel de fila, sin restricciones.
- Las consultas no se ven afectadas por la actualización de las filas.
- Resultados de consultas incluyendo consistencia en lectura.

ALTA DISPONIBILIDAD

- Copia de seguridad *on-line* por fichero, por *tablespace* o por base de datos. A estos tres niveles de arquitectura *Oracle* permite trabajar con copia de seguridad sin necesidad de apagar nuestra base de datos.
- Recuperación *on-line*.
- Ficheros de "log" duplicados.
- Redimensionamiento dinámico y automático de los ficheros de base de datos. Muchas veces cuando un administrador de bases de datos crea una de ellas no está seguro de hasta qué punto debe asignarle espacio libre para prever su futuro crecimiento.

Una mala previsión podía plantearnos serios problemas con las anteriores versiones de este producto. Pero el problema ya está solucionado. Nuestros ficheros de datos pueden ser configurados para que crezcan automáticamente según sea la necesidad de espacio en nuestra base de datos.

- *Oracle Fail Safe:* Exige un conjunto de funciones opcionales, sólo disponible en *NT*.

CONSULTAS Y TRANSACCIONES DISTRIBUIDAS

Ya es posible disponer de sistemas cliente/servidor y *Network Computing* altamente distribuidos, por una fracción del coste y complejidad de los sistemas tradicionales. Las consultas y actualizaciones distribuidas permiten compartir datos almacenados en múltiples servidores garantizando la consistencia de los mismos.

- Consulta y actualización distribuida de forma transparente.
- Validación de la transacción distribuida *Commit*, en dos fases.
- *Joins* distribuidos optimizados automáticamente.
- Transparencia de localización y del protocolo de red.

RÉPLICA DE DATOS

- Realiza múltiples copias de sólo lectura.
- Consistencia e integridad de datos de las transacciones validadas.
- Réplica de tabla completa y parcial.
- Refresco incremental de las tablas replicadas.
- Réplica de datos basada en eventos y por demanda.

RESTRICCIONES DE INTEGRIDAD DECLARATIVAS

- Soporte de restricciones de integridad de entidad declarativa y referencial, 100%.
- Estándar *ANSI/ISO*.
- Restricciones *CHECK*, *DEFAULT*, o *NULL*.
- Declaración de claves Primaria, Ajena, Unica.
- Borrado en cascada.
- Comprobación de restricciones al final de la sentencia *SQL* o de la transacción.

SEGURIDAD DE LOS DATOS

- Autenticación de usuario interna o externa.
- Las opciones externas incluyen el sistema operativo y la



red (*Exige Oracle Advance Networking Option*).
- Imposición de la política de contraseñas
- Usuarios y funciones globales
- Encriptado de datos completa mediante los algoritmos *DES* y *RSA RC4*.
- Privilegios de base de datos
- Seguridad jerárquica basada en funciones

SOPORTE DE LENGUAJES NACIONALES

- Soporte *NLS* completo para 8-, 16- y 32-*bit NLS*, idiomas europeos y asiáticos.
- Variable *Unicode UTF-2* con codificación.
- Conversión de caracteres para entornos heterogéneos.
- Soporte de calendarios.

CARACTERÍSTICAS SQL SERVER 7.0

ALMACENAMIENTO SIMPLIFICADO EN DISCO

Nueva arquitectura de almacenamiento en disco que permite la escalabilidad desde bases de datos de equipos portátiles hasta las empresariales de tamaño de *terabyte*.

En la anterior versión de *SQL Server*, la 6.5 los datos se almacenaban en dispositivos de datos, mientras que en la 7 han sido sustituidos por ficheros del sistema operativo. En un dispositivo de *SQL Server 6.5* podíamos almacenar diversas bases de datos o distribuirlas en diferentes dispositivos. En la nueva versión de *SQL Server*, al igual que en *Oracle*, almacenamos independientemente cada base de datos en un conjunto de ficheros.

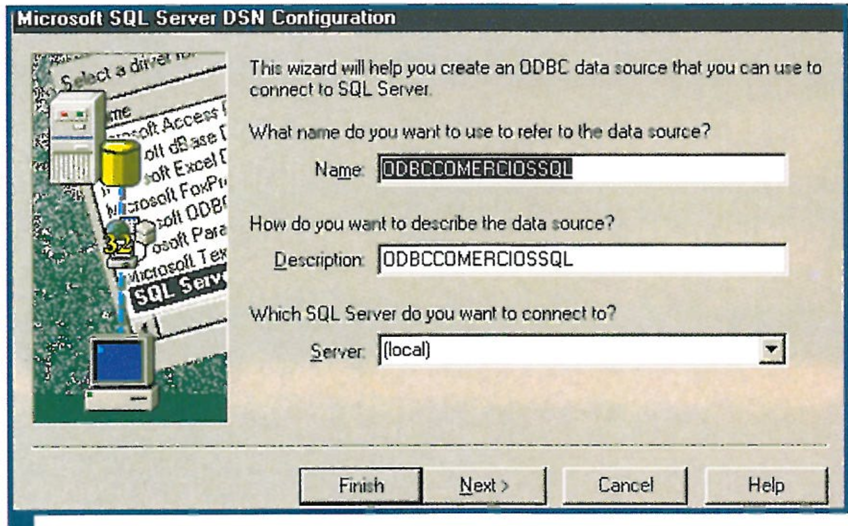


Figura 3.- Primer paso en la creación de la conexión *ODBC*.

OPTIMIZADOR DE CONSULTAS CON MÚLTIPLES FASES

Busca el plan óptimo para consultas para mejorar el rendimiento de consultas complejas. Con el fin de escoger el mejor camino a la hora de resolver nuestras peticiones *SQL*, *SQL Server* analiza la naturaleza de cada consulta para elegir un plan a seguir a la hora de realizar los distintos pasos para devolvernos la respuesta.

SERVICIOS DE TRANSFORMACIÓN DE DATOS

DTS simplifica el proceso de importación y transformación de datos de orígenes múltiples y heterogéneos, tanto de manera interactiva como automática. Se pueden crear objetos de transformación personalizados que se integran en aplicaciones de terceros. Los *DTS* soportan la identificación del origen de los datos, lo que facilita el seguimiento de la procedencia de los datos y el momento en que se produjo.

E/S

Los bloques de *E/S* son cuatro veces mayores que en la anterior versión, las páginas son de **8** *Kb*, las extensiones de **64** *Kb* y los recorridos usan bloques de **64** *Kb*. La *E/S* inteligente es la tecnología clave para mejorar el rendimiento con grandes tamaños de *E/S*. Las lecturas anticipadas más eficientes, los recorridos en el orden físico de las filas y la *E/S* paralela también mejoran el rendimiento general.

ESTRATEGIAS DE ÍNDICES

Una buena estrategia de índices puede ser la responsable de que nuestra base de datos sea considerada excesivamente lenta, o como la más rápida en resolver nuestras peticiones.

Los índices habilitan una mejora significante del rendimiento. Las nuevas estrategias incluyen el uso de varios índices en una única tabla o en varias tablas, índices de cobertura múltiple y combinados, creación de índices paralelos en la misma tabla y mantenimiento automático de estadísticas de manera predeterminada.

COMBINACIONES

Las combinaciones de unas tablas con otras son una de las tareas más comunes de un gestor de bases de

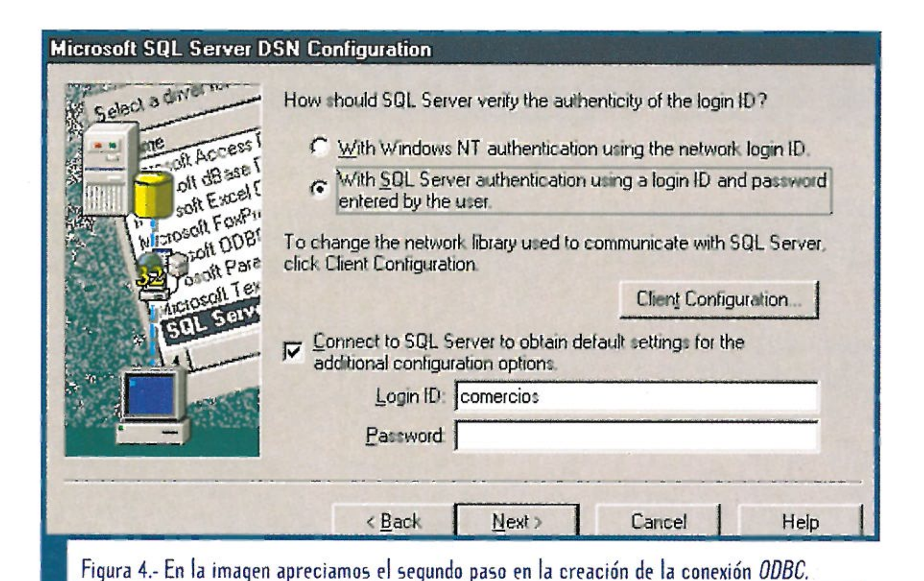


Figura 4.- En la imagen apreciamos el segundo paso en la creación de la conexión *ODBC*.

datos, un punto clave a la hora de identificar un gestor rápido o lento.

SQL Server tiene estrategias de combinación *hash* y mezcla que mejoran el rendimiento de ciertos tipos de recuperación de datos, además de las combinaciones de bucles anidados. En una única consulta se pueden utilizar varios tipos de combinaciones. El procesador de consultas reconoce ciertos tipos de combinaciones comunes.

CONSULTAS PARALELAS

El rendimiento mejora si se utiliza la ejecución en paralelo de consultas de una única consulta a través de múltiples procesadores. Los pasos en una única consulta se ejecutan en paralelo, obteniendo un tiempo de respuesta óptimo. Además, como procesador de consultas admite las bases de datos grandes y consultas complejas encontradas en la ayuda a la toma de decisiones, en el almacenamiento de datos y en las aplicaciones *OLAP*.

ADMINISTRACIÓN DINÁMICA DEL ESPACIO

Las bases de datos pueden crecer o reducirse automáticamente dentro de límites configurables, minimizando la necesidad de la intervención del administrador de bases de datos. Ya no es necesario asignar espacio y administrar las estructuras de datos.

ORDENACIÓN

Las ordenaciones son otra de las tareas más comunes así como más costosas para un gestor de bases de datos. La velocidad de ordenación se ha mejorado ostensiblemente respecto a la anterior versión del producto, especialmente cuando *tempdb* está en un conjunto de discos con bandas.

MEMORIA DINÁMICA

Mejora el rendimiento optimizando la ubicación y el uso de la memoria. Minimiza los conflictos con otros administradores de recursos.

BLOQUEO DINÁMICO DE FILAS

El bloqueo completo de filas se ha implementado tanto para filas de datos como para entradas de índice. El bloqueo dinámico elige automáticamente el nivel de bloqueo óptimo (fila, página, página múltiple y tabla) para todas las operaciones de base de datos.

COPIA DE SEGURIDAD Y RESTAURACIÓN

Las utilidades de restauración y copia de seguridad paralela escalan a las velocidades de dispositivo. Durante la copia de seguridad, totalmente sin desconexión, se mantiene un bajo impacto de los sistemas operacionales y un muy alto proceso de transacciones de servidor.

CANTIDADES MAYORES DE MEMORIA

SQL Server 7.0 *Enterprise* admite direccionamientos de memoria superiores a **4** *Gb*, junto con *Windows NT Server* "5", sistemas que utilizan procesadores *Alpha* y otras combinaciones.

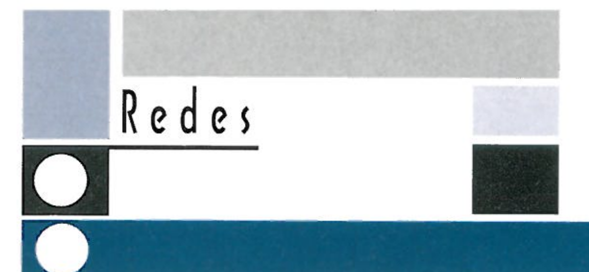
OTROS

Además, *SQL Server* genera estadísticas automáticas mediante el análisis rápido de una muestra, habilitando el Optimizador de consultas para utilizar la información más reciente e incrementar la eficacia de las consultas. También facilita la publicación de datos en la *Web* y proporciona copias de seguridad en línea de alto rendimiento con un impacto mínimo en los sistemas en funcionamiento.

CONCLUSIÓN

Con este artículo finalizamos la serie sobre el desarrollo del cliente y el servidor en nuestra base de datos.

Como se ha podido comprobar a lo largo de estos meses, tenemos la posibilidad de utilizar tanto *Oracle* como *SQL Server*, de tal forma que el lector puede elegir la opción que mejor se adapte a sus características y necesidades.



TRANSFERENCIA DE ARCHIVOS PUNTO A PUNTO (IV)

Enrique de la Lastra, Desarrollador Independiente.

Una transmisión correcta en una comunicación se consigue detectando los posibles errores. Esta tarea es llevada a cabo por el nivel de enlace, por lo que el próximo componente se encargará de la implementación de dicho nivel.

INTRODUCCIÓN

En esta serie, dedicada a la transmisión de datos entre dos dispositivos conectados directamente, hemos implementado hasta ahora un componente que se hace cargo de todas las funciones relacionadas con la transmisión pura de *bits* de información. Además, hemos desarrollado una aplicación que, haciendo uso de este componente, permite transmitir mensajes de texto entre dos ordenadores conectados directamente a través de sus puertos serie mediante un cable cruzado.

Sin embargo, y aunque la aplicación anterior es funcional, ¿qué ocurre si en lugar de conectar los dos *PC's* mediante un cable cruzado los conectamos mediante módem a través de la red telefónica?. En este caso (ver Figura 1), la comunicación ya no será tan fiable como lo pueda ser a través de un cable directo.

La razón es que, por un lado, la distancia física entre los dos ordenadores será mucho mayor y por tanto será mayor la probabilidad de que ocurra un error o un corte de la transmisión. Por otro lado, incorporamos unos dispositivos (*modems*), que transforman la información binaria en señales analógicas que pueden transmitirse por la red telefónica básica. Debido a esta transformación aumenta la probabilidad de encontrar un error en el proceso o de que se produzca un corte en la transmisión.

Si bien los errores de transmisión no serían demasiado importantes en una aplicación tipo *chat* (bastaría reenviar los mensajes que el receptor no haya entendido), sí que resultaría un asunto crítico en una transmisión de ficheros. En este último caso, si se produce un error durante la transmisión de los *bits* de un fichero, el extremo receptor no tendría forma de conocer si lo que ha llegado es correcto o no. Para saberlo, la única vía sería intentar abrir el archivo recibido y comprobar si es o no correcto. De hecho, teniendo en cuenta los formatos propietarios de la mayoría de los archivos de hoy en día, es bastante probable que tras un sólo error en un *bit* del archivo enviado, ni siquiera pudiéramos abrirlo.

En este punto es cuando nos planteamos la implementación de un protocolo que permita asegurar

que lo que recibe un extremo de la comunicación es idéntico a la información que envió el extremo origen. Esta característica, además de muchas otras, se encarga de cumplirla el nivel de enlace. Por tanto, nuestro siguiente desarrollo tendrá como objetivo conseguir un componente que realice las funciones de este nivel, estableciendo una transferencia fiable.

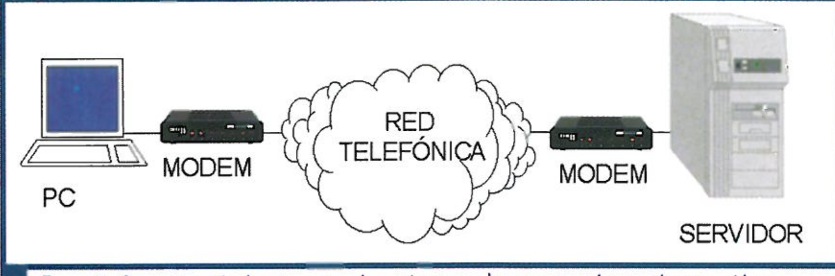


Figura 1.- Comunicación directa entre dos máquinas, cliente y servidor, mediante módem utilizando la red telefónica.

FUNCIONES DEL NIVEL DE ENLACE

El protocolo de comunicaciones de nivel de enlace es responsable de la transmisión fiable de datos sobre un medio serie directo o conectado mediante *módem*. Pero además debe llevar a cabo otras funciones y cumplir con unos determinados requisitos de diseño.

Los principales aspectos que debe cumplir el nivel de enlace, y que deberán quedar cumplidos en la implementación del código, son los siguientes:

- **Transmisión fiable:** El diseño del código debe cuidar los medios de transmisión no fiables, suministrando mecanismos que recuperen la transmisión en caso de cortes o defectos de las líneas.
- **Transparencia:** El diseño debe ser "transparente" con la información transmitida. De hecho, debe entender sólo de *bits*, sin importarle lo que esos *bits* signifiquen para los niveles superiores.
- **Flexibilidad:** El protocolo debe ser independiente de la utilidad práctica a que va a ser destinado. En realidad, esta característica vendrá como consecuencia de la anterior, si se cumple que el diseño es independiente de los tipos de datos que se transmitan por la línea.
- **Independencia de las funciones de los niveles superiores:** El diseño del protocolo debe ser tal que no dependa de las funciones que vayan a implementar los niveles superiores sobre él. Por tanto no debe ocuparse de funciones como son la segmentación de los bloques de información, o el encaminamiento de los datos entre ordenadores de una red.
- **Independencia del nivel inferior:** El protocolo no debe preocuparse de cómo se efectúa físicamente la transmisión de los datos, tan sólo debe utilizar las funciones que el nivel físico le proporciona.

Así pues, en el desarrollo del código, tendremos siempre presentes estas premisas, que asegurarán un correcto diseño.

TRANSPORTE DE LA INFORMACIÓN

Con objeto de cumplir con todos los requisitos de diseño y llevar a cabo las funciones asociadas, la información que se quiere transmitir se agrupa en conjuntos delimitados de *bytes* en los que algunos de ellos se utilizan para analizar el estado de la comunicación. En el nivel de enlace, a estos conjuntos de *bytes* se los denomina tramas (*frames* en inglés).

Los bytes transmitidos se agrupan en tramas de datos delimitadas y predefinidas

Las tramas se componen de dos tipos de datos: los datos de información y los datos de control (Figura 2). Los primeros están compuestos por los *bytes* que el nivel superior entrega para su transmisión, mientras que los datos de control son una serie de *bytes* que añade el nivel de enlace para implementar las funciones antes mencionadas.

Los *bytes* de información provienen del nivel superior y por tanto deben adaptarse a las tramas del nivel de enlace. Es responsabilidad



BYTES DE CONTROL	*BYTES* DE DATOS	*BYTES* DE CONTROL

Figura 2.- Trama de nivel de enlace. Está formada por unos *bytes* de control y otros que contienen la información útil.

del nivel superior enviar información que se adapte a las tramas y no a la inversa.

Todas las tramas que puede transmitir un protocolo de comunicaciones en el nivel de enlace deben quedar perfectamente definidas. De esta forma aseguramos la correcta interpretación de la información en los dos extremos de la comunicación.

ASEGURANDO LA FIABILIDAD

El protocolo de nivel de enlace debe ser fiable, por lo que debe conocer si cada trama ha llegado correctamente. Para alcanzar este objetivo debe verificar dos aspectos: el primero se refiere a que la trama que ha enviado un ordenador haya llegado al otro extremo, ya que se puede haber perdido por un corte en la transmisión o por algún error en los dispositivos de comunicaciones por los que haya atravesado.

El segundo se intenta conocer si los *bytes* recibidos son exactamente los que fueron enviados en el otro extremo. Este segundo caso se puede deber a una pérdida parcial de la información, o al cambio de alguno de los *bits* de la trama, y la causa puede ser el ruido o la mala calidad de las líneas de transmisión.

Para conocer con seguridad que la trama ha llegado al ordenador que se encuentra al otro lado de una conexión punto a punto, cada trama debe ser "asentida" (en inglés: *acknowledged*). Esto quiere decir que cada vez que el extremo receptor recibe una trama, envía de vuelta una trama de *acknowledge* (asentimiento), que le sirve al transmisor de confirmación de que la trama que envío ha llegado al otro lado. A la trama de *acknowledge* se la abrevia comúnmente como trama *ACK* o simplemente *ACK*.

Transmisor
Terminal A
Terminal B
Trama de datos (0)
ACK(0)
Trama de datos (1)
ACK(1)

Figura 3.- Se envía una trama *ACK* cuando se ha recibido una trama y es correcta.

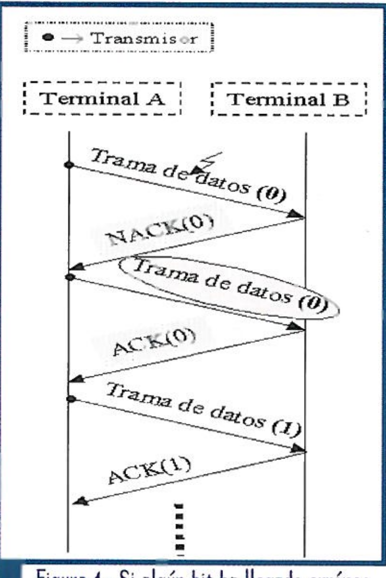


Figura 4.- Si algún bit ha llegado erróneo, el receptor envía una trama *NACK*.

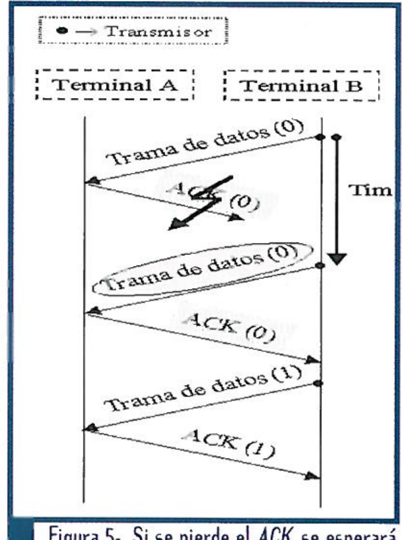


Figura 5- Si se pierde el *ACK*, se esperará un tiempo fijo (timeout) y podremos retransmitir la trama.

La explicación de este proceso se puede apreciar de forma más gráfica en la Figura 3. En ella se representa una comunicación entre dos máquinas conectadas a través de una conexión punto a punto. La primera de estas máquinas está rotulada como *Terminal A* y la segunda *Terminal B*. En sentido vertical y hacia abajo se representa el tiempo transcurrido.

De la transmisión fiable de los datos entre dos extremos se encarga el nivel de enlace

La terminal **A** envía una trama de información numerada como **0.** Cuando la trama llega a su destino, la terminal **B** envía a la línea una trama *ACK* con el número **0.** El hecho de numerar las tramas de datos y las tramas de *ACK* sirve para comprobar que estamos asintiendo la trama correcta. Cuando la trama *ACK(0)* llega a la terminal **A**, ésta sabe que la trama de datos que envió ha llegado correctamente a su destino. Por consiguiente, ya puede enviar la siguiente trama, que aparecerá numerada como trama **1.**

También puede ocurrir que la trama, aunque haya llegado completa al receptor, tenga erróneos uno o más bits. En este caso, en lugar de un ACK se enviará de vuelta al transmisor un NACK (Negative Acknowledge, asentimiento negativo), que advierte de lo ocurrido. En la Figura 4 vemos cómo al llegar la trama NACK(0) la terminal **A** sabe que la terminal **B** ha recibido los datos con error. Por tanto, retransmite la trama de datos número **0.**

Este sistema de confirmación – tanto positiva como negativa - de las tramas, se conoce como "parada y espera" debido a que debemos parar después de transmitir y esperar a que llegue un *ACK* o un *NACK* para continuar transmitiendo.

Este procedimiento de actuación tiene un problema que a su vez tiene una fácil solución. En el caso de que, por un corte en la línea o algo similar, se pierda la trama de *ACK*, la máquina que envió la trama de datos estaría esperando indefinidamente a que llegue ese *ACK*.

Para solucionar esta situación se crea un temporizador denominado: *timeout*, que arranca en el mismo momento en que enviamos una trama. Si pasado este *timeout* no llega una trama *ACK* o *NACK*, deducimos que se ha perdido esa trama y repetimos la transmisión de última trama de información que habíamos enviado a la línea.

En la Figura 5 se puede observar este caso, siendo ahora la terminal **B** el que transmite los datos y la terminal **A** la que envía las tramas *ACK* y *NACK*.

Para saber si una trama que ha llegado completa, lo ha hecho correctamente, se introduce en cada una un par de *bytes* con un código que se genera a partir de los *bits* de la trama. El procedimiento de creación de este código lo define una norma de la *UIT* (*Unión Internacional de Telecomunicaciones*) como *Código de Redundancia Cíclico –CRC-* de 16 *bits*.

El *CRC* lo genera el transmisor utilizando los *bits* de la trama que va a enviar. A continuación añade este *CRC* a la trama y lo envía en conjunto con ella. El extremo receptor vuelve a generar el *CRC* con los *bits* de la trama que le van llegando. Si al terminar la trama, el *CRC* recibido coincide con el *CRC* que se ha enviado en conjunto con la trama, significa que ésta ha llegado correctamente. En otro caso, significará que al menos uno de los *bits* de la trama recibida no es correcto y por tanto, se enviará de vuelta al transmisor una trama de *NACK*.

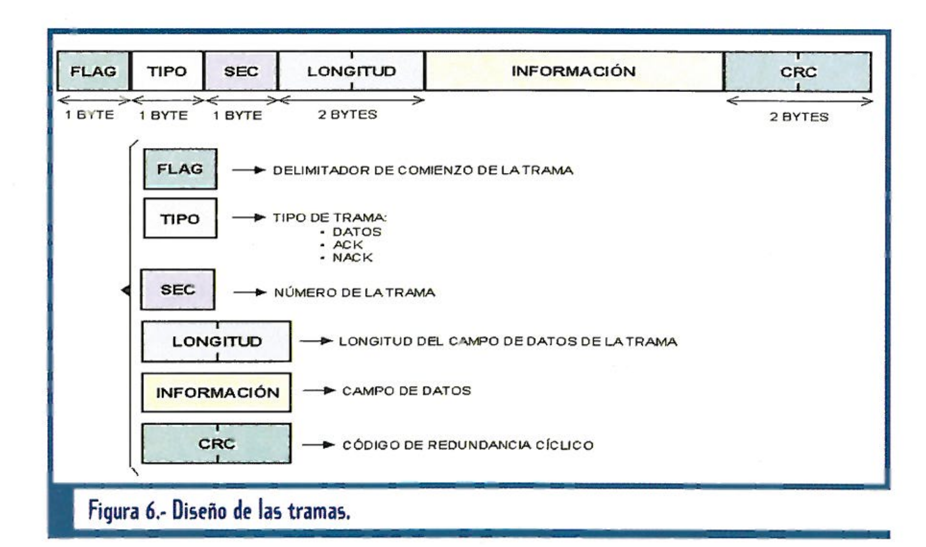


Figura 6.- Diseño de las tramas.

DISEÑO DE LAS TRAMAS

Las tramas, al ser agrupaciones delimitadas de *bytes* deben tener algún mecanismo para distinguir dónde empiezan y dónde acaban. Uno de los mecanismos más comúnmente empleados en el diseño de protocolo de nivel de enlace es el de definir un *byte* específico para indicar el comienzo de una trama y luego utilizar un par de *bytes* que almacenen la longitud completa de la trama. De esta forma podemos deducir fácilmente dónde comienza y dónde termina cada trama.

Al *byte* de comienzo se lo denomina *flag* (bandera). Se suele utilizar el siguiente *byte* como *flag*:

```
01111110
```

En el receptor, cuando llega un *byte* comprobamos si se trata del *flag*. Si no lo es, lo descartamos, y esperamos a que llegue el siguiente. En caso contrario, empezamos a almacenar los *bytes* que vayan llegando, pues sabemos que forman parte de una trama.

Dentro de la trama definiremos un campo especial, formado por dos *bytes*, que determinará la longitud completa de la misma. Como conocemos el comienzo y la longitud, también conocemos dónde termina. Además de esta información incluiremos un *byte* que indique el tipo de trama. Con este *byte* sabremos si estamos transmitiendo una trama de datos o una de *ACK*.

La trama ACK se envía al transmisor cuando los bits recibidos son correctos

Aunque se podrían definir diferentes clases de tramas con distintos tamaños y distintos campos, según el tipo, en nuestro caso vamos a realizar un diseño uniforme, de manera que todas las tramas tengan los mismos campos y lo único que varíe sea el campo de "tipo de trama". Este diseño nos ayudará en el caso de que al mejorar nuestro desarrollo queramos cambiar el protocolo.

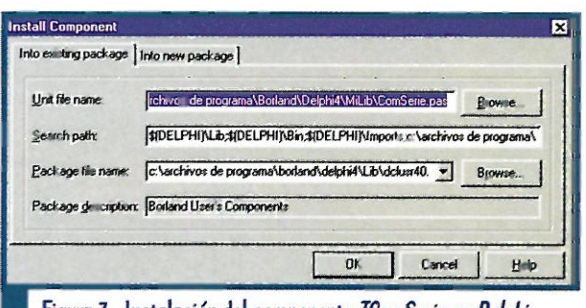


Figura 7.- Instalación del componente *TComSerie* en *Delphi*.

Al final de la trama añadiremos el *CRC*, generado en el ordenador origen utilizando todos los *bytes* de la trama.

El diseño final de las tramas se puede observar en la Figura 6, donde sólo hay que añadir que el campo de información contiene los datos que el nivel superior quiere transmitir (es decir, la información útil de la trama).

IMPLEMENTACIÓN DEL PROTOCOLO

Ya hemos visto cómo va a funcionar el protocolo y hemos definido los *bytes* de control de las tramas así como los tipos de tramas posibles. El único concepto que no se ha desarrollado es el procedimiento de toma del medio de transmisión, es decir, ¿quién empieza a transmitir?

En un principio se podría acordar que en cualquier momento que haya algo que transmitir, se envía sin más a la línea. Si ésta es bidireccional, se podría dar la circunstancia de que al mismo tiempo, el otro extremo podría tomar la misma decisión. Por tanto estaríamos enviando tramas de asentimiento conjuntamente con las tramas de datos (si el medio es unidireccional, esto no ocurre, ya que el primero que empiece a transmitir ocuparía la línea).

Salvando ese caso, lo habitual es que uno empiece a transmitir mientras el otro se encuentra en "reposo". De esta manera, y por simplificar, existe en cada momento un transmisor y un receptor. Aunque el diseño se puede cambiar con un poco más de trabajo para que la comunicación sea bidireccional en todo momento, lo dejaremos como una de las mejoras del desarrollo.

El transmisor, envía tramas de datos y el receptor envía tramas de asentimiento. En cualquier momento, una vez que el transmisor termine de enviar tramas, uno de los dos extremos puede tener algo que transmitir y sin más lo enviará a la línea. En ese momento ese extremo será el transmisor y el otro extremo será el receptor.

Si el CRC es incorrecto se envía al transmisor una trama NACK

Actuando así, sólo tenemos dos estados posibles: o estamos esperando una trama de datos, o estamos esperando una trama de asentimiento (*ACK*). Por tanto, definimos una variable que almacene en cuál de los dos estados nos encontramos:

```
type
  TEstadoTransmision = (e
        etEsperoDatos,
        etEsperoAck);

...

  estadoTx : TEstadoTransmision;
```

El caso que comentábamos antes de que los dos extremos, en una línea bidireccional, se decidiesen a transmitir en el mismo instante de tiempo (y por tanto pasasen al estado de "espera de *ACK*" indefinidamente), se soluciona definiendo un temporizador (similar al *timeout*) de duración aleatoria en cada extremo. De esta forma al vencer los temporizadores, la retransmisión de la trama se produce en instantes distintos en cada extremo de la comunicación. El primer temporizador que venza, será el que defina al transmisor.

TIPOS Y VARIABLES NECESARIOS

Como todo el proceso de desarrollo va a terminar en la creación de un componente *Delphi*, heredado del que hemos creado en los artículos anteriores, lo primero que hacemos es instalar el componente que habíamos creado, en la paleta de componentes.

Seleccionamos la opción **Install Component** del menú **Component.** Aparecerá la pantalla de la Figura 8, donde seleccionaremos la ruta completa donde se encuentra el componente (en este caso la ruta completa es: **C:\Archivos de programa\ Borland\ Delphi\ MiLib\ ComSerie.pas).** Pulsaremos **OK** y aparecerá una pantalla en la que se indica que el paquete **dclusr40.ppl** (en la versión *4* de *Delphi*) se va a recompilar. Una vez que haya terminado el proceso, *Delphi* nos informará de la instalación satisfactoria del componente y desde ese momento ya lo tendremos disponible en la paleta de componentes.

En la Figura 8 se puede ver el componente *TComSerie* en un formulario, añadido directamente desde la paleta, con las propiedades publicadas en el *Inspector de Objetos*.

Para crear el código inicial de nuestro nuevo componente utilizamos la herramienta que *Delphi* suministra a tal efecto. Seleccionaremos **New** en el menú **File**. Se mostrará una ventana con los posibles objetos y aplicaciones que podemos crear. Seleccionaremos **Component** y pulsaremos **OK.**

En la ventana siguiente, elegiremos la clase padre (*Ancestor type*), que en nuestro caso será la clase *TComSerie* que acabamos de instalar. El nombre de la clase será *TComEnlace*, ya que vamos a implementar un componente de comunicaciones en el nivel de enlace.

Después de pulsar **OK**, *Delphi* creará el código inicial del componente, sobre el que empezaremos a trabajar.

Si la comunicación es módem–módem a través de la red telefónica, la transmisión de datos puede sufrir errores

Lo primero que haremos será definir una estructura que contenga las variables de la trama de nivel de enlace. Utilizaremos esa estructura como si de la trama se tratase, por tanto definiremos una para los datos que se van a transmitir y otra para los datos recibidos.

La definición del tipo de datos de esta estructura y de las variables necesarias dentro de la clase, será la siguiente:

Figura 8.- Componente *TComSerie* en *Delphi* una vez instalado y añadido a un formulario.

```
const
// Tamaño máximo del campo de
    datos
// de la trama
MAX_BYTES_FRAME = 200;

TTrama = record
  Tipo   : byte;
  NumSec : byte;
  Length : word;
  Info   : array
    [0..MAX_BYTES_TRAMA+1] of
    char;
  CRC    : word;
end; {TFrameData}

TComEnlace = class(TComSerie)
private

//Estructuras de datos de la
    trama enviada
// y de la recibida }
  TramaRx: TTrama;
  TramaTx: TTrama;

// Estado de la transmisión }
  estadoTx : TEstadoTransmision;
```

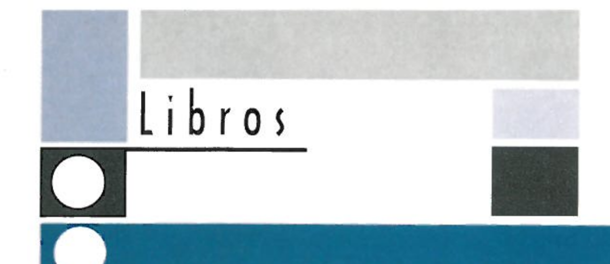
Hemos añadido una constante (*MAX_BYTES_TRAMA*) que permitirá cambiar fácilmente el tamaño máximo del campo de datos de la trama. Además se ha incluido la declaración de la variable **estadoTx** que servirá para conocer en todo momento en qué estado de la transmisión nos encontramos –es decir, si esperamos una trama de datos o esperamos una de asentimiento-.

Y en este punto terminamos para continuar en el siguiente artículo, donde terminaremos el desarrollo de este componente que implementará el protocolo de comunicaciones en el nivel de enlace.

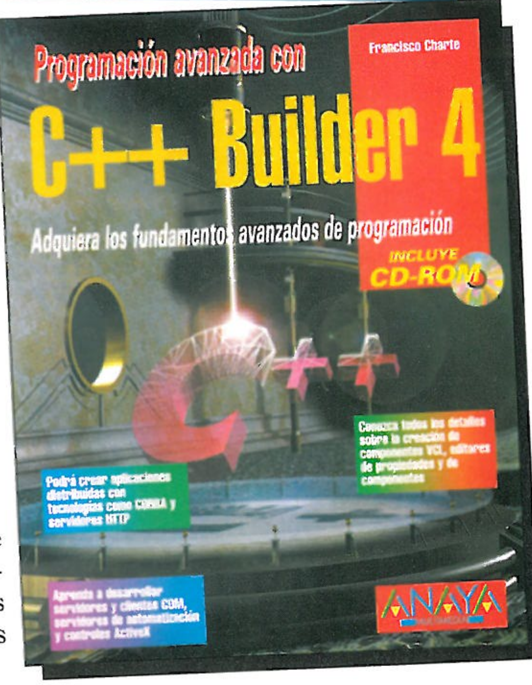
CONCLUSIÓN

En una transmisión de datos punto a punto, uno de los aspectos más importantes a considerar se refiere a la verificación de la transmisión correcta de los *bits* de información. En este artículo hemos visto las funciones que llevará a cabo el componente *software* que va a implementar el nivel de enlace y además, hemos esbozado las líneas principales de su desarrollo.

En los próximos artículos realizaremos la implementación completa de este componente, y de igual forma que en los artículos anteriores de la serie, concretaremos, en una aplicación práctica el desarrollo realizado hasta ese momento.



PROGRAMACIÓN AVANZADA CON C++ BUILDER 4



Con este manual podrá aprender a crear sus propios componentes *VCL*, editores de propiedades y editores de componentes, así como todos los detalles del proceso. Podrá conocer las bases de los dos modelos de componentes más importantes en la actualidad: *COM* y *CORBA*.

El usuario aprenderá a desarrollar servidores y clientes *COM*, servidores de automatización y controles *ActiveX*, podrá crear aplicaciones distribuidas con tecnologías como *CORBA* y servidores *HTTP*. Así mismo llegará a conocer todos los secretos y detalles acerca de la creación de servicios en *Windows NT/2000* y *Windows 9x*. Este libro incluye introducciones al *Component Object Models* y a la *Active Template Library*.

Además de todo esto "Programación avanzada con *C++ Builder* 4", le ofrece la posibilidad de aprender a crear animaciones gráficas con *DirectDraw*, a utilizar acciones propias predefinidas, a realizar las consultas a bases de datos a través de la *Web* o recuperar archivos de diversos servidores *FTP*.

Editorial:	Anaya Multimedia	Autor:	Francisco Charte Ojeda
Nº de páginas:	512	Idioma:	Español
Nivel:	Avanzado	Precio:	5.495 Pts. (I.V.A. inc)

LINUX: RECURSOS PARA EL USUARIO



Los negocios están descubriendo que *Linux* puede proporcionarles computación poderosa, flexible y personalizada a una fracción del costo de los sistemas operativos *UNIX*. En este libro *Jim Mohr* le muestra con detalle cómo usar *Linux* como su principal sistema operativo en el trabajo y en el hogar.

Este ejemplar le proporcionará una introducción completa a las funciones e interacciones en el corazón de este interesante sistema operativo. Hará un completo repaso por los *shells* y *utilerías* de *Linux*, la instalación, administración, supervisión y actualización del sistema, así como la resolución de problemas para aplicaciones de negocios.

El autor ha dado un enfoque especial a la conectividad de redes con *Linux*, la construcción de un servidor *Internet* y todo tipo de posibilidades dentro de la Red.

Empezando por los puntos básicos y continuando con todas las ventajas de este sistema, *Mohr* abre un campo de posibilidades enormemente extenso a los seguidores del potente *Linux*.

Editorial:	Prentice Hall	Autor:	James Mohr
Nº de páginas:	789	Idioma:	Español
Nivel:	Intermedio	Precio:	5.150 Pts. (I.V.A. inc)

tema *Linux* para operar correctamente con esta técnica se pueden encontrar en el *LINUX IP MASQUERADE mini HOWTO.*

La respuesta a tu segunda pregunta depende del *software* que utilices para recibir el correo, así como del sistema operativo de tu servidor.

Sin embargo la manera más sencilla de solucionar tu problema es contactar con el administrador de correo de uno de los sistemas e informarle de tus necesidades. Así el servidor de *mail* modificado enviará automáticamente los mensajes a la dirección común.

Si una de las direcciones pertenece a uno de los proveedores de correo clásicos de *Internet*, busca entre las diversas opciones que ofrecen la de redireccionar todo el correo a una segunda dirección.

La tercera solución sería utilizar en una de tus direcciones un lector de correo programable (con macros), y crear una función que reenvíe los mensajes al recibirlos. Esta solución tiene un inconveniente fundamental, ya que la redirección sólo se produce cuando recogemos los mensajes.

Finalmente, si uno de tus servidores de correo es una máquina *Unix* y tienes acceso vía *TELNET* puedes utilizar el siguiente sistema:

1. Entra en la máquina vía conexión *TELNET*.
2. Crea, en tu directorio raíz un fichero llamado "**.forward**". El punto antes de la palabra *forward* no es una errata.
3. En la primera línea de este fichero introduce tu otra dirección de correo.
4. Una vez realizada esta operación, la máquina enviará a tu segunda dirección todos los *mails* que lleguen a la cuenta de la máquina *Unix*.

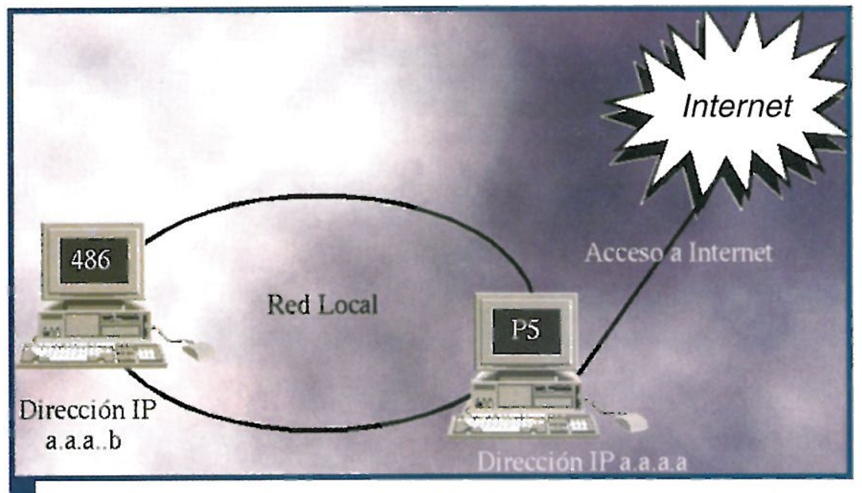


Figura 2.- La alternativa al enmascaramiento de la sub-red es conseguir direcciones válidas para todas las máquinas del sistema.

PREGUNTA

Estimados amigos:

Desde hace ya algún tiempo estoy utilizando un *SGBD* relacional para mantener los datos de los empleados de mi empresa, así como toda la información relacionada con diferentes proyectos.

Actualmente introducimos los datos de una manera manual. Sin embargo, nos gustaría realizar estas inclusiones automáticamente. Nosotros empleamos la versión 8.0 de *ORACLE* como motor de la base de datos, sobre un servidor *Unix*. Entre las diferentes herramientas que están a nuestra disposición se incluyen compiladores de *C, Fortran, Pascal, Prolog* y *COBOL*.

Gracias por sus molestias.

RESPUESTA

Estimado Lector:

Organizar la información de cualquier actividad en la que se genere una gran cantidad de datos es siempre una tarea ardua. El problema se plantea cuando, ya acabada la tarea de introducir los datos, se genera nueva información que ha de ser incluida.

Lo ideal parecería un procedimiento automático de inserción en el motor de gestión de la base de datos. Dependiendo del producto que utilicemos tendremos diferentes herramientas encaminadas a facilitar nuestra tarea.

En concreto, *ORACLE* ofrece varios métodos para construir este tipo de utilidades. Tanto su suite de productos *ORACLE Applications*, encaminados a la gestión de procesos de la empresa a gran escala, como *Developper* 2000 o bien los precompiladores de *C* y *Cobol*, son alternativas válidas para solucionar tu problema.

Si buscamos una solución para cualquier problema de carga o extracción de datos, y además contamos con compiladores de *C* o *COBOL*, lo ideal será utilizar las herramientas basadas en precompilación como pudiera ser el *Pro*C*. Con la ayuda de unas extensiones *SQL* al lenguaje elegido (*C* en el caso de *Pro*C*) podremos acceder directamente a los datos del motor relacional y utilizarlos en nuestras aplicaciones *C*.

ENCUESTA LECTORES

Otoño 1999

Ante todo, desde SÓLO PROGRAMADORES, queremos agradecer el aluvión de encuestas que hemos recibido. Gracias a vuestra colaboración esperamos mejorar día a día la revista. Lamentablemente no tenemos un regalo para todos vosotros. En esta página podéis encontrar a los agraciados con las 70 suscripciones por un año, los 20 teclados Microsoft Natural y los 10 adaptadores Wizard Radio.

70 SUSCRIPCIONES POR UN AÑO A SÓLO PROGRAMADORES



Alfredo Vegas Aceitores (Madrid)
Vicente López de Lerma (Villanueva de la Cañada - Madrid)
Juan Salvador Huertas Romero (Madrid)
Luis Amor Sánchez (Madrid)
Salvador Martínez Palacios (Badalona - Barcelona)
Francisco José Maestre Morera (Oliva - Valencia)
Enrique Cortés Peciña (Madrid)
Belén Arias García (El barco de Valdeorras - Orense)
Sofía Pérez Días (Monforte de Lemos - Burgos)
Itziar Eñero Zabaleta (Azkoitia - Gipuzkoa)
José Manuel Perulero Castaño (Madrid)
Virginia Perulero Castaño (Madrid)
Carmen Cañadilla Jiménez (Madrid)
Jesús Muñoz Lazano (Albacete)
Pedro Gragera Guisado (Madrid)
Rafael Martínez Torres (Segovia)
Daniel Sojo Díez (Abrera - Barcelona)
Francisco Novella Benavent (Valencia)
Sergio García Martín (Cádiz)
Eduardo Cortaberria Vesga (Vitoria - Alava)
Francisco González Serrano (Madrid)
Agustín García González (Salamanca)
Jordi Vila Castaño (Navás- Barcelona)
Juan Luis Korta Bastida (Azpeitia - Gipuzkoa)
Pedro N. Jiménez Mylonás (Ribarroja - Valencia)
Vicente Badenes Donoso (Castellón)
José M. Mejía León (Alcora - Castellón)
Carlos Vigil Montserrat (Badalona - Barcelona)
José Ignacio Fernández Castaño (Piedras Blancas - Asturias)
José Manuel Gómez Soriano (Gandia - Valencia)
José Antonio Fernández Gamez (Colmenar Viejo - Madrid)
Alberto Gil Arpal (Zaragoza)
Leopoldo Colorado Valverde (Las Rozas - Madrid)
David López Muñoz (Madrid)
Inmaculada Gebrie Vicente (Valladolid)
Carlos Alvarez Diab (Alcalá de Henares - Madrid)
José María Arrabal Alcañiz (Don Benito - Badajoz)
Pedro Gabriel Salazar Nascimiento (Leganés - Madrid)
Miguel Izquierdo Martín (Valladolid)
Iván García Santamaría (Vitoria - Alava)
Eduardo Casas Izquierdo (Logroño - La Rioja)
Ricardo Muñoz Fernández (Zaragoza)
José Fernández Sánchez (Alcantarilla - Murcia)
Francisco Javier García Oncala (Sevilla)
Alberto Vegara Cerezo (Orihuela - Alicante)
Bernardo Torres Gabaldó (Valencia)
Jaume Ucher Parra (Barcelona)
Francisco Serrano Pina (Murcia)
Luis Escalona Parrilla (Sevilla)
Antonio Albarrán Cabrera (Olvera - Cádiz)
Gerard Martí Escapa (Pineda de Mar - Barcelona)
Alberto Ruiz García (Burgos)
B. Calzaverini (Valencia)
Jesús J. Catalán Ramírez (Tomelloso - Ciudad Real)
Estíbaliz Gallardo Martín (Málaga)
Vicente José Martínez Balaguer (Alicante)
Igancio Rodríguez García (Sevilla)
Pedro López Juvez (Madrid)
Rafael Fernández Mejías (Fuenlabrada - Madrid)
Fernando Javier García Alemany (Villanueva de la Torre - Guadalajara)
José Luis Pastor López (Badalona - Barcelona)
José María Vázquez Pedraza (Coin - Málaga)
Juan Francisco Fernández Carrasco (Barcelona)
Felipe del Amo Mazario (Barcelona)
Samuel Zarza Fernández (Madrid)
José Ignacio Auditore Brichete (Talavera de la Reina - Toledo)
José María Osuna Romero (Sevilla)
Joseba Rubio Galdós (Pallejá - Barcelona)
Susana Isabel Gancedo Guindos (Fuentesnuevas - León)
Vicente García Armero (Balsicas - Murcia)

20 TECLADOS MICROSOFT NATURAL

Rubén López Valderrama (Madrid)
Jorge Sanz de Acedo Sebastián (Burgos)
Alberto Valero Gómez (Madrid)
Alberto Vidal Panalés (Cartagena - Murcia)
Mario Antón Peña (Madrid)
José Fernando Moya Burón (Guadalajara)
Angel Bescos Hernández (Zaragoza)
Pedro Cañadilla Jiménez (Madrid)
Susana Carcayo Ruíz-Andino (Santander - Cantabria)
Juan Carlos Lagoa Serrano (Mérida - Badajoz)
Eugenio Alvarez Martínez (Barcelona)
José Ricardo Colina Pérez (Madrid)
Ginés Vidal Bravo (Palma de Mallorca - Baleares)
Antonio Llin Ferrero (Ontinyent - Valencia)
Jesús María Chamizo Carmona (Fuenlabrada - Madrid)
Ramón Sanz Martínez (San Sebastián de los Reyes - Madrid)
Juan Manuel Manrique Martín (Alcobendas - Madrid)
Mª Isabel Rodrigo Rodrigo (Alcorcón - Madrid)
Julián Rodríguez Gil (Pontevedra)
Luis Miguel Cuenca Sauce (Madrid)

10 ADAPTADORES WIZARD RADIO



Juan Antonio Criado Redondo (Barcelona)
José Luis Rando Calvo (Málaga)
Ana Belén Zotes Calvete (Madrid)
Daniel Costas Rodríguez (Burgos)
José Javier Carro Palomo (Pozuelo Estación - Madrid)
Gustavo Fernández Baillo (Madrid)
Abel Hernández Valero (Zaragoza)
Juan de Dios Martínez Ortega (Almería)
Francisco Villegas Martín (El Ejido - Almería)
Francisco López Bolivar (Granada)


Microsoft Windows
¿Preparado
para
desarrollar...